

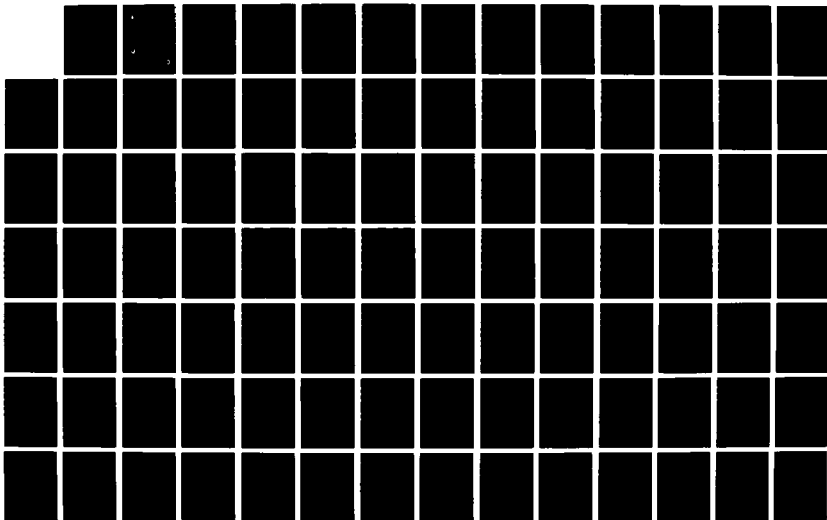
AD-A100 002

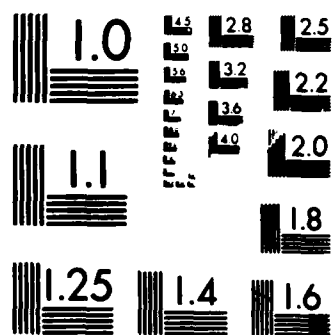
FEASIL IMPLEMENTATION UNDER VAX VMS WITH DESIGN  
INFORMATION(U) ALABAMA UNIV IN HUNTSVILLE DEPT OF  
ELECTRICAL AND COMPUTER EN. J D MARR ET AL NOV 86  
UAH-5-31325 AMSHI-CR-RD-55-86-5 F/G 12/5

1/4

UNCLASSIFIED

NN





MICROCOPY RESOLUTION TEST CHART  
NATIONAL BUREAU OF STANDARDS 1963-A

2

AD-A180 002



TECHNICAL REPORT CR-RD-SS-86-5

FEASIL IMPLEMENTATION UNDER VAX VMS WITH  
DESIGN INFORMATION

J. D. Marr, B. E. Tucker, T. A. Palmer,  
D. C. Pool, and T. N. Long  
The University of Alabama in Huntsville  
Huntsville, Alabama 35805

NOVEMBER 1986

Prepared for:  
System Simulation and Development Directorate  
Research, Development, and Engineering Center

Contract No. DAAH01-82-D-A008



**U.S. ARMY MISSILE COMMAND**

*Redstone Arsenal, Alabama 35898-5000*

Approved for public release; distribution is unlimited.

DTIC  
ELECTE  
MAY 01 1987  
S R E D

87 4 30 247

REPORT DOCUMENTATION PAGE				Form Approved OMB No J704 0188 Exp Date Jun 30 1986	
1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b RESTRICTIVE MARKINGS		
2a SECURITY CLASSIFICATION AUTHORITY			3 DISTRIBUTION / AVAILABILITY OF REPORT		
2b DECLASSIFICATION / DOWNGRADING SCHEDULE			Approved for public release; distribution is unlimited.		
4 PERFORMING ORGANIZATION REPORT NUMBER(S) 5-31325			5 MONITORING ORGANIZATION REPORT NUMBER(S) CR-RD-SS-86-5		
6a NAME OF PERFORMING ORGANIZATION School of Engineering		6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION System Simulation and Development Directorate		
6c ADDRESS (City, State, and ZIP Code) The University of Alabama in Huntsville Huntsville, AL 35805			7b ADDRESS (City, State, and ZIP Code) Commander US Army Missile Command ATTN: AMSMI-RD-SS-SE Redstone Arsenal AL 35898-5252		
8a NAME OF FUNDING / SPONSORING ORGANIZATION		8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DAAH01-82-D-A008		
8c ADDRESS (City, State, and ZIP Code)			10 SOURCE OF FUNDING NUMBERS		
			PROGRAM ELEMENT NO	PROJECT NO	TASK NO 0043
					WORK UNIT ACCESSION NO
11 TITLE (Include Security Classification) FEASIL IMPLEMENTATION UNDER VAX VMS WITH DESIGN INFORMATION					
12 PERSONAL AUTHOR(S) J. D. Marr, B. E. Tucker, T. A. Palmer, D. C. Pool and T. N. Long					
13a TYPE OF REPORT Final		13b TIME COVERED FROM 12/6/84 TO 5/31/85		14 DATE OF REPORT (Year, Month, Day) November 1986	
15 PAGE COUNT 368					
16 SUPPLEMENTARY NOTATION					
17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP	FEASIL		
			Database Management System		
			FORTRAN		
19 ABSTRACT (Continue on reverse if necessary and identify by block number) Implementation of the FEASIL data base management system under a VAX VMS operating system is described. Technical considerations are reviewed, and new features that resulted and code changes that were required are summarized. Improvements in the SORT function are also described and quantified. A significant amount of FEASIL design information is presented in the form of flow charts for the entire body of code.					
20 DISTRIBUTION / AVAILABILITY OF ABSTRACT <input type="checkbox"/> UNCLASSIFIED/UNLIMITED <input checked="" type="checkbox"/> SAME AS RPT <input type="checkbox"/> DTIC USERS			21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED		
22a NAME OF RESPONSIBLE INDIVIDUAL M. M. Hallum, III			22b TELEPHONE (Include Area Code) (205) 876-4141		22c OFFICE SYMBOL AMSMI-RD-SS-SVSS



## PREFACE

This technical report was prepared by the Research Staff of the Electrical and Computer Engineering Department, School of Engineering, The University of Alabama in Huntsville. The purpose of the report is to provide documentation of the technical work performed and results obtained under delivery order 0043 of MICOM Contract No. DAAH01-82-D-A008; Terry N. Long, Principal Investigator. The report also provides an addendum to the documentation of the technical work performed and results obtained under delivery order 0027; Dr. James D. Marr, Principal Investigator. The project was performed by Dr. James D. Marr, Bruce Tucker, Tim Palmer, David Pool, Terry Long, and several other members of the UAH ECE research staff. Dr. M. M. Hallum III, Chief, System Evaluation Branch, was the technical monitor; and Mr. Mark Horton also from the Systems Evaluation Branch of the Research, Development, and Engineering Center, U. S. Army Missile Command, provided technical coordination.

The technical viewpoints, opinions, and conclusions expressed in this report are those of the authors and do not necessarily express or imply policies or positions of the U. S. Army Missile Command.

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Avail and/or	
Dist	Special
A-1	



# TABLE OF CONTENTS

	<u>Page</u>
1.0 INTRODUCTION .....	1
2.0 VAX IMPLEMENTATION TECHNICAL CONSIDERATIONS .....	2
3.0 NEW FEATURES .....	4
4.0 CODE CHANGES .....	5
5.0 REVISED SORT PROGRAM .....	10
5.1 Introduction .....	10
5.2 Revised SORT Technique .....	10
5.3 SORT Improvement Quantification .....	11
6.0 FEASIL Design Summary .....	12
7.0 CONCLUSIONS AND RECOMMENDATIONS .....	13
REFERENCES .....	25
APPENDIX A - MAIN .....	A-1
APPENDIX B - RETEVD .....	B-1
APPENDIX C - DSPLAY .....	C-1
APPENDIX D - RLFUNC .....	D-1
APPENDIX E - PRINTREL .....	E-1
APPENDIX F - SORT2 .....	F-1
APPENDIX G - EDTREL .....	G-1
APPENDIX H - NEWRELAT .....	H-1
APPENDIX I - DELREL .....	I-1
APPENDIX J - MODCOL .....	J-1
APPENDIX K - MERGRL .....	K-1
APPENDIX L - RORGRL .....	L-1
APPENDIX M - BKUPRL .....	M-1
APPENDIX N - DBSTAT .....	N-1
APPENDIX O - HELP .....	O-1

# TABLE OF CONTENTS (CONT'D)

	<u>Page</u>
APPENDIX P - TPL1B1 .....	P-1
APPENDIX Q - TPL1B2 .....	Q-1
APPENDIX R - TPL1B3 .....	R-1
APPENDIX S - TPFILECR .....	S-1
APPENDIX T - F710 .....	T-1
DISTRIBUTION .....	Dist-1

## 1.0 INTRODUCTION

The FEASIL Database Management System (DBMS) is a relational system; that is, it is designed to most effectively support tabular data. It was first developed about 1978 by M. M. Hallum and has evolved several new features since that time. Currently, it can accept new data from keyboard or file, alter existing data, select only certain parts of the data for further manipulation, sort data into ascending or descending order, perform limited mathematical functions, display or plot data, print tables of data, output data for use by other programs, and perform several other functions.

FEASIL was first developed under a Perkin-Elmer OS-32 operating system and has since received many refinements through several revisions. The principal task of the work documented by this report was to transfer FEASIL to a VAX-11/780 computer under a VMS operating system. Section 2.0 describes several of the technical considerations which affected the work. New features which resulted from the transfer are then described in Section 3.0. Code changes in addition to the new features were required for the transfer. They are documented in Section 4.0. Major performance improvements resulted during the transfer with the SORT function receiving the greatest improvements. These improvements are described and quantified in Section 5.0. As part of this and previous tasks, an effort was made to carefully examine the body of FEASIL code through development of flow charts. This design summary effort is described in Section 6.0, and the flow charts are presented in the appendices. And finally, an extensive list of recommendations and conclusions is presented in Section 7.0.

## 2.0 VAX IMPLEMENTATION TECHNICAL CONSIDERATIONS

The principal goal of this project was to port FEASIL from a Perkin-Elmer (P-E) computer to a VAX computer. Differences between the P-E and VAX that made this conversion nontrivial included file structure differences, different file name limitations, Flex dialect differences, and operating system differences. Within the main goal was the requirement that the transported version be upward compatible and have substantially similar capabilities. A secondary goal was to improve feedback to the user.

From the programmer's viewpoint, the P-E files reside directly on a device, with pathnames and filenames such as

```
USER:FILE.EXT
```

typical. On the VAX, the device is subdivided into directories and possibly subdirectories, with a name such as

```
dual:[userdisk1.humanbeingname.activityname]filename.extension
```

typical. The device name and directory path for a first level user subdirectory is typically 28 characters. Further, files in the "current default directory" of the VAX can be accessed without either the device name or the directory path name. This forces creation of a large buffer for device name (including path) and the option to have no device name.

Filenames on the P-E are limited to 8 characters with a 3-character extension, as noted above. The VAX filenames are limited to 39 characters with a 39-character extension. The easiest solution is to just continue use of the 8+3 filenames. Since the directory name is also limited by the same 39-character limit, and operator tolerance is the principal limit on the number of levels of subdirectory, the device name buffer and any descendant buffers should be huge. As a practical limit, we assumed 44 characters as an upper bound for name and path.

The most annoying difference between the Flex dialects was that the construct

```
WHEN(I.LT.0)
    I=0
    GO TO 73
ELSE
    I=1
FIN
.....
```

converts without error on the P-E, while it produces code on the VAX of the form

```
IF(.NOT.(I.LT.0))GO TO 99999
I=0
GO TO 73
GO TO 99998
99999 I=1
99998 .....
```

This causes a FORTRAN compile error because the statement "go to 99998" is unreachable. A more significant problem is that some of the more powerful FORTRAN 77 (F77) constructs (such as the "if...then...else...endif") have keyword conflicts with Flex, making them incompatible. However, Flex can understand only capital letters; making all F77 constructs lower case lets them pass the flecs preprocessor unmodified and be compiled.

Operating system differences became most apparent in file operations. The two computers differ in their mechanisms for renaming files, but that problem can be overcome within the FILEREN subroutine itself. As expected, I/O error numbers were incompatible, so a separate routine called VAXERR was created and linked through the original error handler. Tape handling is sufficiently difficult that the backup function was forced into a stand-alone program.

### 3.0 NEW FEATURES

New features visible to the user fall into two categories, flexibility and feedback. The first group includes menu modifications. The second group consists of messages to inform the user of current activity and of the progress of long-duration activities.

The main menu originally had functions 0 to 11 (function 12 is limited to system programmers). The VAX version has the same choices, but will also accept commands by the first letter(s) of the command name; the user need enter only enough letters for a nonambiguous choice. This is of benefit to casual users and novice typists who can reach letters more easily than digits. Another system program function (13) has been added to perform a brute-force tuple dump to the printer; this is effective for archival dumps and for debugging. Help options are available at the main menu and at several of the secondary menus.

FEASIL now provides feedback approximately every twenty seconds or less. When the original .ADF and .TF are being copied at the start of "retrieve, manipulate, and plot" (8), the program declares that the copy is beginning and reports every hundred records copied. The SORT function now provides a crude advance estimate of the time needed and periodically reports how the bubble sort is progressing. The DELETE function in EDIT (4) now reports every twentieth record for large delete actions.

Additional feedback information was added to several of the error messages and to dangerous activities. When a FORTRAN error occurs, instead of giving machine-dependent error numbers, the system now generally gives a text string defining the error. Since the MOVE (M) command in RETRIEVE, MANIPULATE AND PLOT can delete tuples from the original relation, the user is now warned a second time that he has made a dangerous choice and asked to reconfirm the decision.

#### 4.0 CODE CHANGES FROM PERKIN-ELMER VERSION

Changes to the program were made for two reasons, machine factors and program enhancement. Differences in the file structure were the main reason for the machine-dependent changes, and an attempt was made to confine these to the F710 library. Program enhancement changes included message changes, conversion from Flex to FORTRAN 77, and algorithm changes. Errors found in the flowcharting project were corrected on both the VAX and P-E versions; the errors were predominantly evolutionary in origin, frequently being related to the string-in-TF data compression method. The structure of the .TDF, .ADF, and .TF files was maintained for compatibility with the P-E FEASIL. The rest of this section will provide a brief description of the changes to each program file.

Runtime memory usage was considered first. Currently, the main menu driver and all five libraries are permanently resident in memory, with the ten main-menu options overlaid. A secondary overlay is used for some of the activities under "RETRIEVE, MANIPULATE, AND PLOT" (option 8). Specifically, only one of FUNCTION, PLOT, PRINT, or SORT will be in memory at a given time. Table 1 shows that while many subprograms are called by all ten activities, some are called only by one of the main menu choices. An analysis of this subprogram calling hierarchy indicates that a restructuring of the libraries would be both possible and beneficial. The subprograms and their locations are listed in Table 2, and Table 3 shows the same information from the opposite perspective, location and contents. A file-by-file analysis of changes follows.

P-E files have read/write keys, but these are not available on the VAX; FEASIL generally does not ask for keys on the VAX. A more general rewrite would add a function in the machine-dependent library F710 to request or not request keys.

The file BKUPSTUB (BACKUPRL on the P-E) contains the tape backup routine. Since operating system restrictions on the VAX make the traditional backup method extremely inconvenient, the backup routine, BACKUPRL, has been converted to a "stub" explaining how to call the current version of the stand-alone input program for tapes generated by FEASIL on the P-E. It is suggested that backup on the VAX be performed by standard VAX procedures. Efforts are in progress to write a program to prepare tapes for return to the P-E. One of the smaller compatibility problems is that the VAX and P-E disagree on the byte order for integer numbers on a tape; they agree on text order. Additionally, the VAX insists on specialized system calls for tape data transfer rather than standard FORTRAN READ and WRITE statements.

The file DABA08 contains both the main program and a BLOCK DATA initialization section. The main program is predominantly F77, which more effectively supports the text selection of desired activity. The main menu handler now accepts letters as well as digits; only upper case is supported to remain consistent with the following menus in the original P-E version. All subprogram calls have been modified so that the array length is also passed for any array sent to the subprogram. In the future, it will be possible to dimension all arrays in subroutines to the correct



length activate runtime subscript checking, and eliminate most references to the LCONST COMMON; this will aid in discovering runtime errors that can corrupt data. The COMMON's in the BLOCK DATA section have been adjusted to a minimum length of 19 words to accommodate the possibility of longer file-names on the VAX. The new COMMON named MAXLEN has been added to indicate the longest allowable name, which is set at 13 words at the present. The default device/directory name has been set to 0 characters with text of "Current System Default Directory"; the FILDES routine (elsewhere) was also tampered to be able to echo this default despite its zero length.

The file DABAEXIT contains the status reporting routine DBSTAT. It originally supported the normal status function (11 or S) and the system programmer function (12 or P ). It has been modified slightly toward F77, but the principal change was the addition of a second system function TOTAL (13 or T), which does a total dump of the entire relation to the printer. Damaged or defective ADF records are reported, rather than system failure resulting. Suppression of this second system function could be accomplished most easily by modification of the main menu to reject the new commands. The function to be performed is now specified by a character variable passed as the final parameter. The modifications were also retrofitted to the P-E version.

The file DELETREL contains the relation deletion routine DELREL. It is still in Flex, but is only 46 lines long including comments. No changes were necessary.

The file DSPLAY contains the plotting routine DSPLAY, originally written by M. Castellano based on earlier code by J. Marr. It was changed only slightly and still relies heavily on Flex. The printer device number was changed from 13 to 6. The pen plotter function is still a nonfunctioning stub. It could stand some modification for compactness and efficiency since it reimplements some functions available in the libraries.

The file EDITREL7 contains the data entry and editing routine EDTREL. Few changes were made in the conversion process, but several logic errors were discovered during flowcharting and were corrected on both the VAX and P-E versions. The most visible changes were periodic feedback in the add-data-from-a-file function (A), the search/find function (F), and the delete function (D); all now comment on progress about every 20 seconds.

The file F7IO is a machine-dependent library. All routines have builtin error checking and messages. Subroutine FILEOPN uses the standard OPEN function for 'OLD' and 'DIRECT'. Subroutine FILEDEL on the VAX opens a file and then closes it with 'DELETE' option; this may be portable for future use. Subroutine CREAFILE checks for file prior existence (OPEN as 'OLD') before creating the file by using an OPEN as 'NEW' and 'DIRECT' with specified record length and file length. The check for prior existence is because the VAX allows creation of a new version of an existing file without producing an error message. Subroutine FILEREN required extensive rewriting. The name of the file on the input logical unit is determined using an INQUIRE; the logical unit is closed; the file is renamed by a system library function; and the file (under either the new or old name) is reOPENed on the logical unit. Subroutine FILECLS consists of only the

FORTRAN CLOSE function. Subroutine MODAP is a stub with no executable statements. Finally, subroutine VAXERR accepts an error number, prints to the screen its meaning (if known), and reports back whether a meaning was printed.

The file FUNCTION contains the mathematical functions routine RLFUNC used in option 8. It was modified to run faster by simple rearrangement of some of the code. A logic error was also detected that caused an infinite loop for partially active data; this was corrected in both versions. It still uses Flex.

The file HELP contains the help routine HELP.

The file MERGERL contains the relation-merging routine MERGRL. It now provides a progress report every 20 lines during the add (A) function.

The file MODCOLUM contains the column-modification routine MODCOL. It was substantially unchanged, and still relies heavily on Flex.

The file NEWRELAT contains the relation-creation routine NEWREL. It was changed only slightly and still relies heavily on Flex.

The file PRINTREL contains the printing routine PRINTREL used in option 8. It was changed to account for text length differences. It still relies heavily on Flex.

The file REORGREL contains the relation-reorganization routine REORGRL.

The file RETRIEV7 contains the retrieve/manipulate/plot function RETEVD, which is option 8. Most modifications were implemented to improve readability of the code or feedback to the user, but some logic errors were also eliminated. A warning and request for verification were added to the delete-tuples-after-move option on the move command (M). It relies heavily on Flex.

The file SORT2 contains five routines in support of the sort function used in option 8. The CSORTF routine builds a file containing the tuple number and the tuple as text data from the designated column; it was not changed and still relies heavily on Flex. The SORTFL routine was substantially rewritten and details appear in the next section. The supporting routines SORTFA, ASCTYP, ASCCHR are new and include only two instances of Flex.

The file TPFILECR is a library of routines related to file creation. Unless noted, changes are for readability and feedback only. Subroutine FILDES inquires directory name (calling it "volume name"), verifies legality, updates the default volume name if appropriate, and builds an extended name of volume and filename and period. FILDES needed substantial modification to handle the changes in names on the VAX. The handling of the "default directory" message was ad hoc, and should be changed in later revisions. SECURE gets the read/write keys, but is not called on the VAX; it could be replaced by a stub. EMIARY just writes zeros into an array; it

now consists of a DO loop. ADFCRE, TDFCRE, and TFCRE add the appropriate extension to the base relation name and create the file via CREAMFIL. RELFC just calls these three routines. ADFDEL, TDFDEL, and TFDDEL add the appropriate extension to the base relation name and delete the file via FILEDEL. RELFD just calls these three routines. EQUIVI copies N words from one array into another; it now consists of a pretest and a DO loop. OPNADF, OPNTDF, and OPNTF add the appropriate extension to the base relation name and open the file on the specified logical unit. FILERR checks for recognized FORTRAN error numbers, but is somewhat machine dependent. NAME gets a relation name of not more than 42 characters, which is later "hashed" to make the file name. All of these routines were originally Flex, but some have been altered to remove some of the vestiges. The OPNTF routine originally used a FORTRAN COMMON area for building the extended name. However, the RETEVD function, which calls OPNTF, used the same COMMON for another purpose, and the conflict resulted in system malfunctions. As a result, OPNADF, OPNTDF, OPNTF, ADFCRE, TDFCRE, TFCRE, ADFDEL, TDFDEL, and TFDDEL were modified to eliminate use of the COMMON. This change to local arrays caused no change in the size of the .obj file and reduced the size of the .exe (executable) file.

The file TPLIB1 is a library of text and number manipulation routines. Few changes were necessary in this file, and most routines rely heavily on Flex. STRLEN reports the length of a string and copies it to FEASIL format. FTIPI converts a floating point (real) number to pseudo-integer by saving it to memory as "real" and reading it back as an integer; an EQUIVALENCE installed in a rewrite about 1984 simplified the code and made it portable. PITFP converts pseudo-integer to real by the same approach. YESNO demands from the user an answer of YES or NO (Y or N); it still uses FEASIL string manipulation techniques. Subroutine GETNUM gets text in alpha-numeric form from the user and converts it via REANUM. REANUM converts text to the corresponding integer value. ELEB eliminates leading edge blanks by shifting the FEASIL format text left. ENFPTA converts real numbers to FEASIL string format. ENINTA does the same for integers. GETFP converts a text number from the terminal into floating point. REAFP does the actual conversion. All of the routines in this library use the GET routine to read characters from the terminal.

The file TPLIB2 is a library of routines for file access. Few changes were necessary; about half of the routines are pure F77. Routines PUTTDF and RTVTDF open the TDF file via OPNTDF, write or read the file, respectively, and close the file via FILECLS. ADDTUP adds a tuple to an existing relation. DELTUP removes a tuple. PUTIFO puts data into the .ADF or .TF as appropriate for the data strategy. PTFRC and PADFCRC write data to the .TF and .ADF files, respectively. If the string sent to PADFCRC is shorter than 8 characters and an override has not been declared, then PADFCRC will return a signal and two words to be stored in the TF. RTFCRC and RADFCRC read data from the .TF and .ADF files respectively. COLNAM gets the name of a specified column from the .ADF file. HASHIT computes the file name by hashing the relation name. The first four letters of the hash name are the same as for the relation name; the fifth is a length signal; the sixth and seventh are a hash of relation name characters 5 to 42; and the eighth letter is 'A'. It was modified recently (1985) to use an algorithm which

reduces hashing collisions; the P-E version checks to see if a file from the previous version exists and recommends conversion. UTFAR converts the two TF words into a 1 to 7 character FEASIL format string into the ADF buffer, inverting the TF storage mode of PADFRC.

The file TPLIB3 is a library of routines for file maintenance. RDOADF and RDOTF create a larger .ADF/.TF file, copy the old information into it, and delete the old version. CPYREL copies both the .ADF and .TF files into a new name. FSLIDE shifts an array downward one word and is used to convert FEASIL format text into FORTRAN format. Only slight changes have been made from the original Flex text.

Some additional tools have been built as part of the conversion process. NAMES examines an existing relation, reports the long name, and prints a macro to rename the relation to the new hash code from either the original "hash" or the original name. QUICK is a VAX program to convert a tape-transferred relation from P-E tape format (copied to the disk first using MOUNT/FOREIGN and COPY) to a standard relation. The command file which does the copy and calls QUICK is named FEASILTAPE. A relation repair function named PATCH is under development.

## 5.0 REVISED SORT PROGRAM AND TIMING TESTS

### 5.1 Introduction

Users have long complained of the slowness of the SORT function, and it has been speculated that this is due to the frequent disk accesses of the file-to-file algorithm. When an attempt was made to run the original algorithm on the VAX, the machine malfunctioned. It was then decided to modify the routine. The original sort program operated in two phases, (1) copying of the tuple number and the data from the designated column to a sort file SF in CSORTF and (2) sorting of the data in SORTFL. These two routines were called in sequence by the RETEVD routine. In order to preserve modularity, all changes were made internal to the SORTFL routine.

### 5.2 Revised SORT Technique

Three acceleration measures were instituted simultaneously. They were an in-memory routine for small data sets, a different buffering scheme for other sorts, and code tightening. The new SORTFL routine first builds file names and opens the SF to a logical unit. If the strategy is nonstring and the tuple count is less than 1000, it calls SORTFA to do the in-memory sort and then returns. Otherwise, it creates two scratch files 11 and 12. If the text is nonstring, the records of these files are only two words wide. The SF is then copied into 11, and the data is sorted using 11 and 12 as "ping-pong" buffers. When the sort is completed, the data is transferred back to the SF as text, the files are closed (and scratch files deleted), and the program returns.

The sort algorithm now used is a weaving bubble sort. Sorting starts at the bottom, and the largest value is carried to the top, remembering when the last swap occurred. Next, from the point of the last swap, the smallest value is carried to the bottom, again remembering the last swap point. The swap points are the new limits for later passes, and execution can stop when they meet. The sort code reports the number of data values swapped every 100 passes to maintain user confidence that the program is still running.

The fast in-memory sort SORTFA also uses the weaving bubble sort, but does not need scratch files. It was decided, for code compactness, to have only one sort sequence and make the order decision based on strategy. The use of logical variables and logical assignment statements and comparisons also aided compactness.

Two auxiliary functions were added to aid the sort. ASCTYP returns an integer value to indicate whether the character input is (1) a blank, (2) a capital letter, (3) a lower case letter, (4) a digit, or (5) something else. This appeared to be close to the original sequence, but there was some evidence that digits should precede letters instead. If some other sorting sequence is desired in the future, this routine could be modified easily. ASCCHR accepts two character values c1, and c2, and returns a logical value of .TRUE. if c1 is less than or equal to c2. ASCCHR uses ASCTYP to separate nonsimilar characters.

### 5.3 SORT Improvement Quantification

The new version appears to require less memory. The rewrite reduced the .FLX file from 953 to 884 lines, and the derived .FOR (FORTRAN) file from 648 to 500 lines. The .OBJ file which contains resulting assembly code and linkage information was reduced from 32 to 31 blocks, a negligible difference but still a reduction.

The major difference is in speed. The new algorithm requires about 10 seconds for the CSORTF conversion and un-conversion, and 23 seconds to sort 500 random floating point numbers (263 passes). The old algorithm took 10 + 600 seconds. This is a factor of 20 to 25 speed improvement. For larger data sets, the operating system handles buffering and the nonstring records are packed many to a disk block, instead of singly. This results in fewer disk accesses and greater speed of execution. String data appeared to sort slightly faster, but this is hard to measure on a multiuser system.

## 6.0 FEASIL DESIGN SUMMARY

A major effort was initiated under a previous task to complete examine the body of FEASIL code through flow charting it. This effort greatly assisted the transfer process through identification of dormant errors in FEASIL code. The flow charts are presented in Appendices A through T. They should prove to be a valuable tool in the analysis of errors still hidden in the VAX implementation. They should also assist in the refinement process that will inevitably follow this effort.

## 7.0 CONCLUSIONS AND RECOMMENDATIONS

Limited testing reveals no errors, so the system appears ready for production work. As with any large software system, problems will appear from time to time and can be treated as they are revealed. Recommendations for future effort can be divided into two areas, enhancement and rewrite.

Several functions could be added to the FEASIL DBMS to enhance its power. The most powerful function would be a mathematical function to compute a linear or nonlinear function of two columns and store the result in a third column. Small changes that might aid the user would include modifying the HELP function to loop, offering its menu again, until the user selects "quit."

Small invisible code changes which would improve performance include (1) increasing the buffer size in SORTFA and (2) when deleting tuples in the edit function, delete the one with the highest number first. The current tuple delete function removes the lowest numbered one and shifts all tuples above it downward; it then deletes the next one. If a relation is 100 tuples long and tuples 11 to 100 are being deleted, then tuple 11 is removed, all 89 remaining are shifted, the new tuple 11 is removed, all 88 remaining are shifted, .... In the same case with the suggested change, no shifts would occur and thousands of disk accesses would be eliminated.

FEASIL has evolved into a powerful tool. It would be desirable to eventually port FEASIL to the HP 9000, the IBM PC, and several other computers. However, the flowcharting and VAX conversion efforts have revealed some problems that impact on portability.

FEASIL was originally written in Flex on a machine subject to frequent malfunctions. Several compromises were made as a result.

(1) Most file accesses (read or write) are preceeded by an open and followed by a close; the overhead and disk accesses of these opens and closes can slow down a program significantly.

(2) The string handling capabilities of Fortran 77 were not available, and several activities were performed in a difficult fashion. For example, a 16-character string can be shifted left one character in F77 by

```
str = str(2:16) ,
```

or the eighth letter of a file name can be changed from 'A' to 'B' by

```
file(8:8) = 'B'
```

Other similar operations are as simple. HASHIT was written using the substring capability shown and is much more compact than the functionally equivalent Flex code would be.



(3) As in any evolving program, array lengths changed with time and arrays were declared as IDENT(1) in subprograms; this forced use of the LCONST COMMON and variables for the constants from 0 to 30 to bypass the automatic subscript checking of FORTRAN.

(4) Ad hoc changes have crept into the code, and nesting has resulted in redundant or contradictory testing for certain conditions. For example, two consecutive blocks of code are conditioned on the same test in DBSTAT rather than merging the blocks and only testing once.

(5) The Flex loop construct appears frequently. The code

```
I = 0
REPEAT UNTIL (I.EQ.N-1)
    A(I+1) = 0.
    I = I + 1
FIN
....
```

results in FORTRAN code similar to

```
      I = 0
      GO TO 99999
99998  IF(I.EQ.N-1)GO to 99997
99999  A(I+1) = 0.
      I = I + 1
      GO TO 99998
99997  ....
```

rather than the F77 code of

```
      DO 10 I = 1, N
10      A(I) = 0.
```

which will probably compile to more compact and faster assembly language code.

(6) Some of the routines, such as EDTREL, have evolved to over 1000 lines. Even with the use of Flex macrofunctions, this is a larger amount of code based on the same global variables than the average programmer can grasp. Reduction of program unit size and more use of subprograms would simplify the code.

(7) Several functions implemented as Flex macros occur in multiple top-level routines. For example, both EDTREL and RETEVD (main menu options 4 and 8) ask for a column number. This is just a request for a number within a certain range, and could be implemented trivially as an integer FORTRAN FUNCTION in a library.

(8) Several routines, such as TFDEL, ADFDEL, and TDFDEL differ only in one location. A generic routine would reduce code space by nearly a factor of 3; the routine could even be called by the existing routines, each of which could be reduced from 24 to 5 lines.

(9) Flowcharting revealed that several routines frequently appear in sequence. A new routine implementing the entire function would shorten the calling program and improve readability. For example, a new routine EXTEND could call EQUIVI, CATSTR, and FSLIDE to create an extended name from the specified base name and specified extension. This would cost one 6-line routine but reduce dozens of triple calls to single highly readable calls.

(10) Given one machine, it was not obvious which routines were really machine dependent, and such functions occur in many places. For example, the read/write keys should be requested in a machine-dependent subprogram. Similarly, all questions could be asked through a subroutine QUEST which prints the test without a carriage return (if the machine allows such an option). A logical function QUESYN could call QUEST to provide the text and then call YESNO to demand a yes or no answer.

A complete top-down restructuring of the code is recommended. Guidelines would include (1) pure portable Fortran 77 code consistent with the standard, (2) copious use of comments in the code, (3) routines rarely longer than 2 pages including the imbedded comments (header comments do not count), (4) addition of new functions/routines whenever appropriate, and (5) restructuring of the libraries to minimize runtime size and to place machine-dependent code in as few files as possible. Implementation of guideline 1 alone would result in an estimated code reduction as shown in Table 4; the quantitative benefits of the other guidelines have been estimated.

Two additional recommendations are related to "user-friendliness". They are (6) any potentially dangerous or time-consuming activity should offer a 'punt' option, and (7) new algorithms should be considered for time-consuming activities. The code should also allow graceful exits from accidentally chosen paths, lest the user be stuck in an infinite question-answer loop that can only be exited by damaging the relation or by killing the program and losing all previous work. Similarly, any activity, such as sort, which could take hours to perform should offer a graceful exit after providing a time estimate; the user may prefer an unsorted list now rather than a sorted list in three hours (or even fifteen minutes). The rewrite of SORT was a start toward algorithm improvement, but room exists for more improvement.

Implementation of FEASIL on the VAX has greatly improved the usefulness of the system. Reduced execution time and more reliable data handling has greatly increased its acceptance by users. Continued refinement and elimination of any errors that may remain should guarantee user acceptance in the future.

TABLE 1 SUBPROGRAM CALLS OF FEASIL ROUTINES

```

name    called by (*=directly, digit=indirectly)

newrel  * !          main menu routines (one routine per file)
delrel  * .!
edtrcl  * ...!
modcol  * ...!
mergrl  * ....!

rorgrl  * ..... !
retevd  * ..... .!
bkuprl  * ..... ..!
dbstat  * ..... ...!
help    * ..... ....!

rloadf  1 ...*. .... !          edtrcl support routines (library routines)
rdotf   1 ...*. .... .!
putifo  1 ...*. .... ..!
cpystr  1 ...*. .... ...!

rlfunc  1 ..... *. ... .. !          retevd support routines (separate)
dsplay  1 ..... *. ... .... .!
printrel ..... *. ... .... ..!
csortf  1 ..... *. ... .... ...!
sortfl  1 ..... *. ... .... ....!
asctyp  2 ..... ..... ..... *t
ascchr  2 ..... ..... ..... *t
sortfa  2 ..... ..... ..... *t
cpyrel  1 ..... *. ... .... ....!          (all routines below)
                                           ( are in libraries )

ftipi   1 ...*. *. ... .. t          format conversion routines
pitfp   1 ...*. *. ... .. *.**.. t
enfpta  1 ...*. .l... .. *. ... !
eninta  1 ...*. .l... .. *. ... .!
utfar   1 ...*. **... .. *.**.. .t
eleb    2 ...ll. .... **.. ..... **!

name     1 ***** ****. .... ..... !          file identification and
hashit   1 ***** ****. .... ..... ..!          confirmation routines
fildes   1 *l*** .***. .... *. ... .. ..!          (fildes also called by relfd)
yesno    1 ***** .***. .... *. ... .. ....!

MAIN      rorgrl      rlfunc      name
           retevd      dsplay      hashit
newrel    bkuprl      printrel     fildes
delrel    dbstat      csortf       yesno
edtrcl    help        sortfl
modcol                      cpyrel
mergrl    rloadf
           rdotf      enfpta
           putifo     eninta
           cpystr     eleb

```

TABLE 1 SUBPROGRAM CALLS OF FEASIL ROUTINES (continued)

name called by (\*=directly, digit=indirectly)

relfc	1	*....	.....	.....	.....	.....	.....	!	relation/tuple operations	
relfd	1	*....	.....	.....	.....	.....	.....	!		
addtup	1	..*.*	**...	.....	.....	.....	.....	..!		
deltup	1	..*..	*....	.....	.....	.....	.....	..!		
colnam	1	*.***	**.*	....	*....	....	.....	.....!		
adfcrc	1	1.11*	***..	*...	.....*	...	.....	*.... !	adf/tdf/tf operations	
tdfcrc	1	1..*	***..	....	.....	...	.....	*.... "		
tfcre	1	1.1.*	***..	*..	.....*	....	.....	*.... "		
adfdel	1	*1...	**...	....	.....	...	.....	*.... .!		
tdfdel	1	*1111	*111.	....	.....	...	*..	..... ."		
tfdel	1	*1...	**...	....	.....	...	.....	*.... ."		
									adf/tdf/tf operations	
radfrfc	1	1.*1*	***1.	....	.l....	...	.....	* .. !		
rtvtdf	1	..***	****.	....	.....	...	.....	.. .!		
rtfrfc	1	..***	****.	..*	.....	...	...	..**.. .!		
padfrfc	1	*****	****.	....	.....	...	.....	.. .!		
puttdf	1	*.***	****.	....	.....	...	.....	.. .!		
ptfrfc	1	..**1	11*..	..*	.....	...	.....	..**.. .!		
									adf/tdf/tf open	
opnadf	2	11111	1112.	*...	.2***.*	...	.....	1 .. *..*.. !		
opntdf	2	1.111	1111.	....	.....	...	.....	.. *..*.. !		
opntf	2	..111	111..	*1.	***.*	...	.....	..11. .. *..*.. !		
									file ops and misc	
creafil	1	..221	1*1..	11..	...**1	...	11...	* .....	. t	
filecls	1	11*11	1**1.	**1.	*****	...	..111	.. ***.*	. t	
fileren	1	..11.	*....	**..	.....	...	.....	.. .....	. m	
filerr	1	12113	11*3.	**2.	.111.1	...	.1..	**222	* 111111 * t	
emiary	1	*1*1*	***1.	**2.	*1*1.1	...	*1*	11222	** 111111 * t	
equivi	1	11*11	**11.	**2.	*1*1.1	11*	*1..	11222	** 111111 * t	
filedel	1	1.11.	**22.	**..	...**.	...	.1..	.1...	* .....	. !
fileopn	1	22222	1*33.	112.	*****1	...	.2..	.2222	.1 111111	* *t
MAIN	rorgri	rlfunc	name	-cre	opn-					
	retevd	dsplay	hashit	-del						
newrel	bkuprl	printrel	fildes		filedel					
delrel	dbstat	csortf	yesno		radfrfc					
edtrcl	help	sortfl			rtvtdf					
modcol		cpyrel	relfc		rtfrfc					
mergrl	rdoadf		relfd		padfrfc					
	rdotf	enfpta	addtup		puttdf					
	putifo	eninta	deltup		ptfrfc					
	cpyst	eleb	colnam							

TABLE 1 SUBPROGRAM CALLS OF FEASIL ROUTINES (continued)

name called by (\*=directly, digit=indirectly)

```

getfp 1 ..*.. .*... .. keyboard input
reafp 1 ..*.. .l... .. *!
getnum 1 *,**.. **... *** .. !
reanum 1 l.*l. .ll.. .... 11l... .. 1**!
getch 1 11*11 1*11. 11.. l.*... 11. *.** .. 1*1*!
get 1 *1*** 1*11* .... 11*... .. *.** .. *.**..!

streq 1 11*** .**1* .... ..l... .. ..* .. 1*.... ! string ops
strlen 1 *1*** 1*11* .... 11*... .. *.** .. *.**.. !
cpysub 1 11*1* **11. **.. *.**.. 11* *.** .. 1*.... !
catstr 1 11*11 **11. **.. *1***1 ... **. 11222 ** 111111 * .. !
catsub 1 22*21 1122. 11.* 121112 221 .111 22333 11 222222 1 . 21.... **!
fslide 1 ..11. .*... **.. *.**.. .. .. !
chtyp 1 11*11 1*111 11.. 111*.* 11* *.11 .. 1*1*.. *.**..!

```

MAIN	rorgrl	rlfunc	name	-cre	opn-	streq
	retevd	dsplay	hashit	-del		strlen
newrel	bkuprl	printrel	fildest		filedel	cpysub
delrel	dbstat	csortf	yesno	radfrc		catstr
edrel	help	sortfl		rtvtdf	getfp	catsub
modcol		cpyrel	relfc	rtfrc	reafp	fslide
mergrl	rdoadf		relfd	padfrc	getnum	chtyp
	rdotf	enfpta	addtup	puttdf	reanum	
	putifo	eninta	deltup	ptfrc	getch	
	cpystr	eleb	colnam		get	

Note 1: The symbol to the right of each line indicates the type of routine:

! calls others  
t terminal routine (calls no others)  
m machine-dependent (only system calls or assembly language)

Note 2: Some routines are included in the source, but not called:

```

catnum      (calls catstr, putnum)
closef
filrenb
hash
modap
newno
opnrel
put  (calls catsub, cpystr, cpysub, putch, putnum)
putch
putnum      (calls putch)
secure      (calls getnum)(called by fildest on the P-E)
strlt

```

TABLE 2 SUBPROGRAM LOCATIONS FOR FEASIL ROUTINES

name ----	file (location) -----	name ----	file (location) -----
MAIN	daba08		
blockdata	daba08		
addtup	tplib2	mergr1	mergerel
adfcrc	tpfilecr	modap	f7io (unused)
adfdel	tpfilecr	modcol	modcolumn
ascchr	sort2	name	tpfilecr
asctyp	sort2	opnadf	tpfilecr
bkupr1	backupr1	opntdf	tpfilecr
catstr	sc	opntf	tpfilecr
catsub	sc	padfrc	tplib2
colnam	tplib2	pitfp	tplib1
cpyrel	tplib3	printrel	printrel
cpyst	sc	ptfrc	tplib2
cpysub	sc	putifo	tplib2
creafil	f7io	puttdf	tplib2
csortf	sort2	radfrc	tplib2
dbstat	dabaexit	rdoadf	tplib3
delrel	deletrel	rdotf	tplib3
deltup	tplib2	reafp	tplib1
dsplay	dsplay	reanum	tplib1
edtrel	editrel7	relfc	tpfilecr
eleb	tplib1	relfd	tpfilecr
emiary	tpfilecr	retevd	retreiv7
enfpta	tplib1	rlfunc	function
eninta	tplib1	rorgrl	reorgrel
equivi	tpfilecr	rtfrc	tplib2
fildeg	tpfilecr	rtvtdf	tplib2
filecls	f7io	secure	tpfilecr (unused)
filedel	f7io	sortfa	sort2
fileopn	f7io	sortfl	sort2
fileren	f7io	strlen	tplib1
filerr	tpfilecr	tdfcrc	tpfilecr
fslide	tplib3	tdfdel	tpfilecr
ftipi	tplib1	tfcrc	tpfilecr
getfp	tplib1	tfdel	tpfilecr
getnum	tplib1	utfar	tplib2
hashit	tplib2	yesno	tplib1
help	help		

Note: Library SC is part of Flex (alternate spelling Flecs); we cannot alter the code for these routines.

TABLE 3 FILE CONTENTS FOR FEASIL ROUTINES

file	contents (subprograms)				
----	-----				
backuprl(P-E)	bkuprl				
bkupstub(VAX)	bkuprl				
daba08	MAIN	block data			
dabaexit	dbstat				
deletrel	delrel				
dsplay	dsplay				
editrel7	edtrel				
f7io	creafil	filecls	modap	vaxerr(vax only)	
		filedel			
		fileopn			
		fileren			
function	rifunc				
help	help				
mergerel	mergrl				
modcolum	modcol				
newrelat	newrelat				
printrel	printrel				
retreiv7	retevd				
reorgrel	rorgrl				
sc	catstr	catsub			
	cpystr	cpysub			
sort2	csortf	sortfl	ascchr	asctyp	sortfa

TABLE 3 FILE CONTENTS FOR FEASIL ROUTINES (continued)

file	contents (subprograms)				
----	-----				
tpfilecr	adfcrc adfdel tdfcrc tdfdel tfcrc tfdel	relfc relfd	opnadf opntf opntdf	emiary equivi filerr	filides name secure
tplib1	eleb enfpta eninta	ftipi pitfp	getnum getfp reanum reafp	strlen	yesno
tplib2	addtup deltup	padfrc putifo puttdf	radfrc rtfrc rtvtdf	colnam hashit utfar	
tplib3	cpyrel	fslide	rdoadf rdotf		



TABLE 4. CURRENT SIZES AND ESTIMATED REWRITE SIZE

file ----	subprogram -----	lines now total code -----		estimated lines in F77 -----	%change -----
daba08	MAIN	159	87	84	3-
	block data	96	70	62	11-
dabaexit	dbstat	330	218	203	6-
deletrel	delrel	46	27	24	10-
dsplay	dsplay	685	505	450	10-
editrel7	edtrrel	2213	1298	1147	12-
f7io	creafil	53	29	28	
	filecls	12	4	4	
	filedel	33	15	14	
	fileopn	52	20	18	
	fileren	43	24	24	
	modap	4	3	3	
	vaxerr	169	161	161	
	total	377	253	249	1-
function	rlfunc	310	197	157	21-
modcolum	modcol	607	306	234	24-
newrelat	newrel	469	264	230	13-
printrel	printrel	1048	725	629	13-
retreiv7	retevd	1780	1007	891	12-
sort2	asctyp	28	11	10	10-
	ascchr	34	16	16	0
	csortf	180	106	92	13-
	sortfa	165	100	94	4-
	sortfl	448	260	250	4-
tpfilecr	adfcrc	32	12	15	
	adfdel	40	19	22	
	emiary	13	5	5	
	equivi	15	7	7	
	fildes	89	53	48	
	filerr	52	47	10	
	name	64	31	23	
	opnadf	53	20	23	
	opntf	53	20	5	
	opntdf	53	20	5	
	relfc	62	38	33	
	relfd	59	25	22	

TABLE 4. CURRENT SIZES AND ESTIMATED REWRITE SIZE (cont.)

file	subprogram	lines now		estimated	Xchange
----	-----	total	code	lines in F77	
	secure	34	30	30	
	tdfcre	32	12	5	
	tdfdel	40	19	5	
	tfcre	32	12	5	
	tfdel	40	19	5	
	total	802	389	268	34-
tplib1	eleb	46	28	14	
	enfpta	53	32	27	
	eninta	26	11	10	
	ftipi	15	6	6	
	getfp	35	14	14	
	getnum	29	8	8	
	pitfp	13	6	6	
	reafp	225	144	20	
	reanum	156	89	20	
	strlen	78	37	21	
	yesno	59	29	24	
	total	774	404	170	58-
tplib2	addtup	59	20	14	
	colnam	24	6	6	
	deltup	47	18	12	
	hashit	61	43	43	
	padfrc	103	67	66	
	ptfrc	35	11	11	
	putifo	83	43	37	
	puttdf	40	21	17	
	radfrc	56	25	22	
	rtfrc	46	17	16	
	rtvtdf	38	18	14	
	utfar	47	22	21	
	total	668	311	279	10-
tplib3	cpyrel	108	68	59	
	fslide	26	8	8	
	rdoadf	135	87	76	
	rdotf	139	83	76	
	total	426	246	219	11-

## OVERALL TOTALS

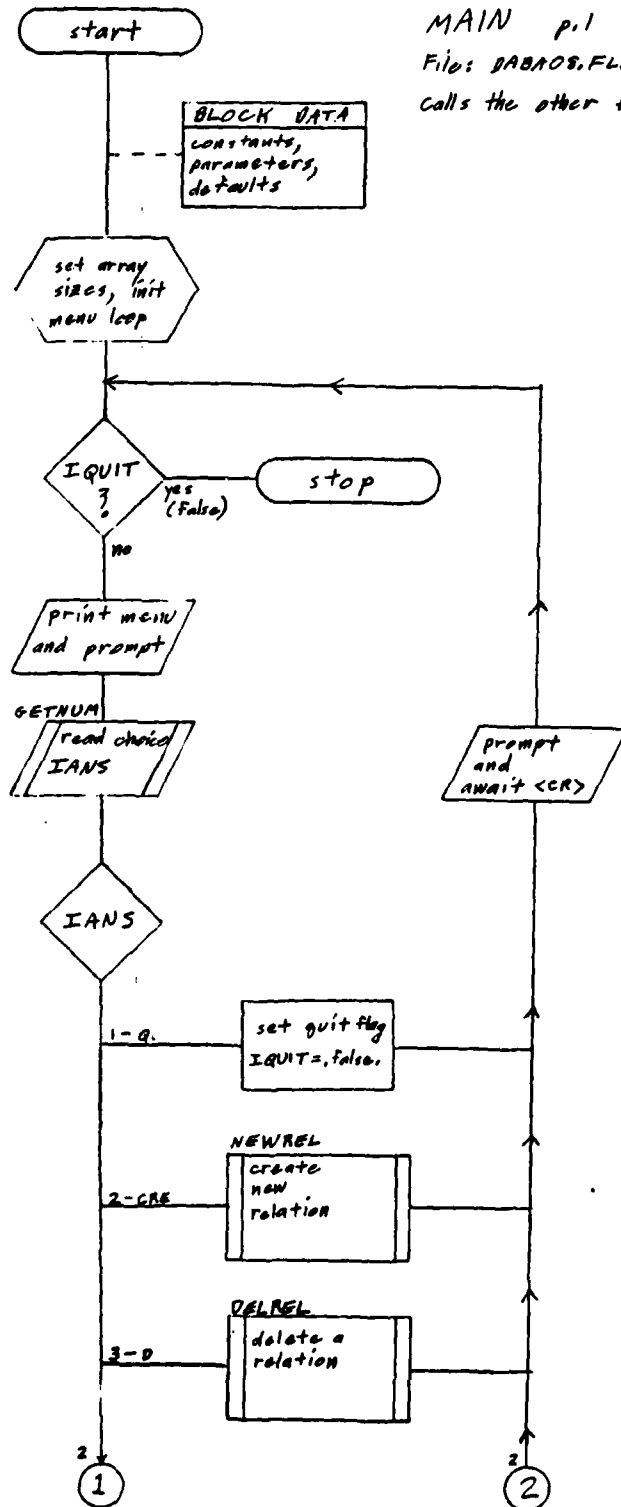
total lines including comments	10,845	(38% comments)
total lines code	6,800	
estimated lines if FORTRAN 77	5,758	
estimated code reduction	1,042	
estimated code reduction percentage	15%	

## REFERENCES

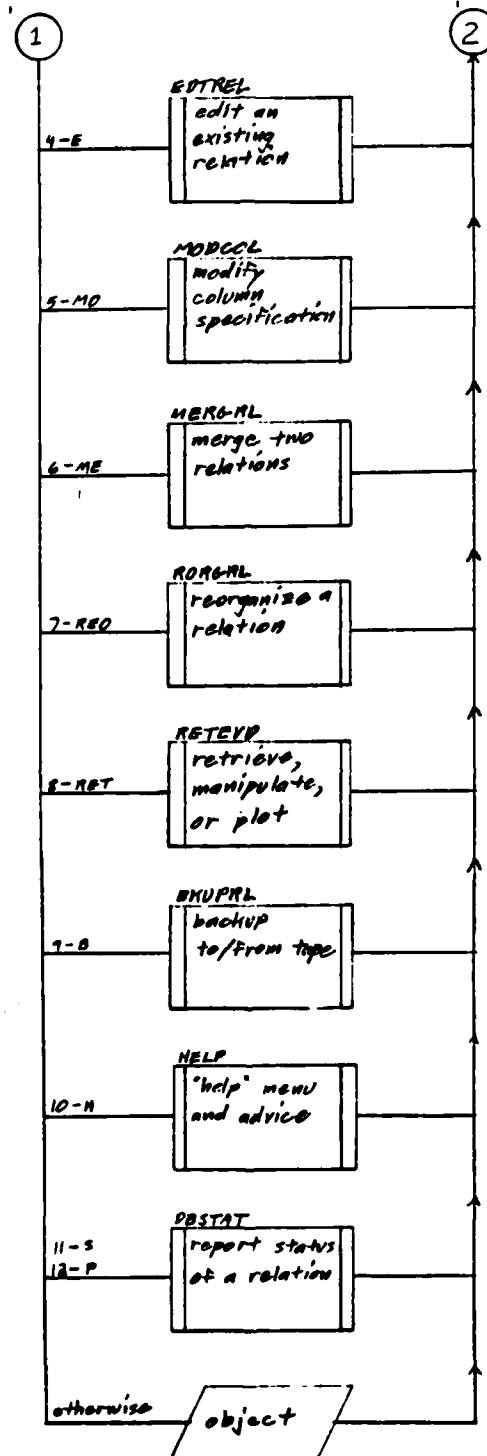
- (1) Technical Report 83/11, "Data Base Support for SIO," James D. Marr, Laura Pritchett, and Bill Wells, School of Engineering, University of Alabama in Huntsville, March 1984, Contract DAAH01-82-D-A008 Delivery Order #0015.
- (2) Technical Report RD-80-11, "A Relational-Based Data Management System for Engineering and Scientific Application," Maurice M. Hallum III, Systems Simulation and Development Directorate, U.S. Army Missile Laboratory, June 1980.
- (3) Technical Report 324, "Software Development for the Analysis of Improved Hawk Weapons System," Gerald Johnson, Laura Jeziorski, James Marr, School of Engineering, University of Alabama in Huntsville, July 1982, Contract DAAH01-82-D-A006 Delivery Order #0005.
- (4) Technical Report 5-31303, "FEASIL User's Guide," Lisa M. Clay, Bruce E. Tucker, and Terry N. Long, School of Engineering, University of Alabama in Huntsville, December 1985, Contract DAAH01-82-D-A008 Delivery Order #0038.

APPENDIX A

MAIN



MAIN p.1  
File: DABA08.FLC  
Calls the other functions.



APPENDIX B

RETEVD

Subroutine: RETRIEVE

This subroutine allows the user to output or manipulate relations. In outputting a relation the user may:

1. Print data in a relation
2. Retrieve data
3. Create a relation from data
4. Plot data to screen, plotter, or printer

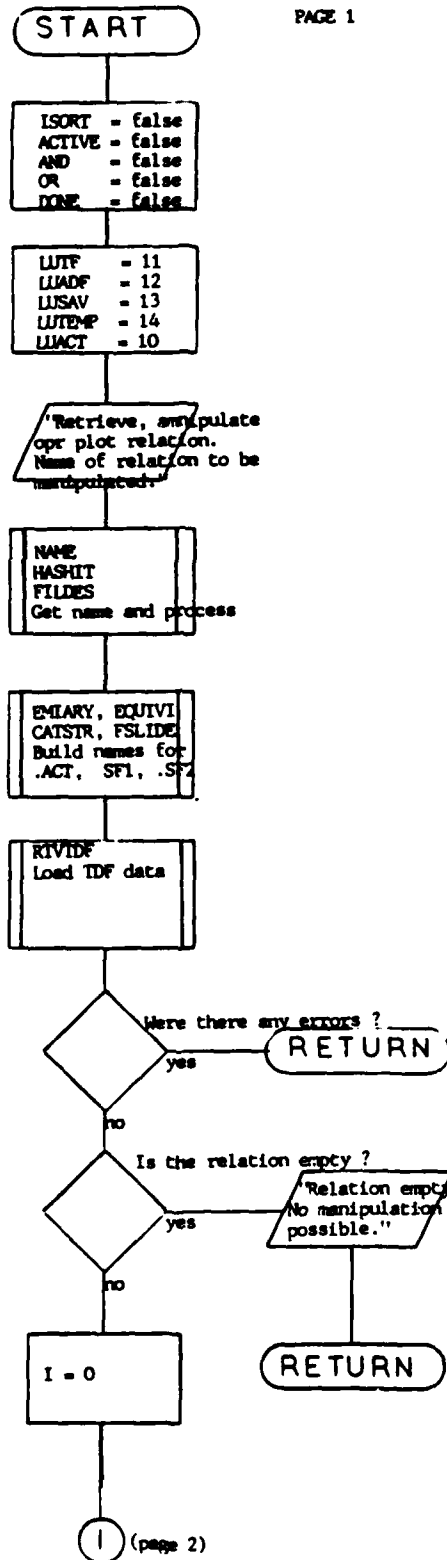
In manipulating a relation the user may:

1. Sum columns
2. Average columns
3. Calculate the mean and standard deviation



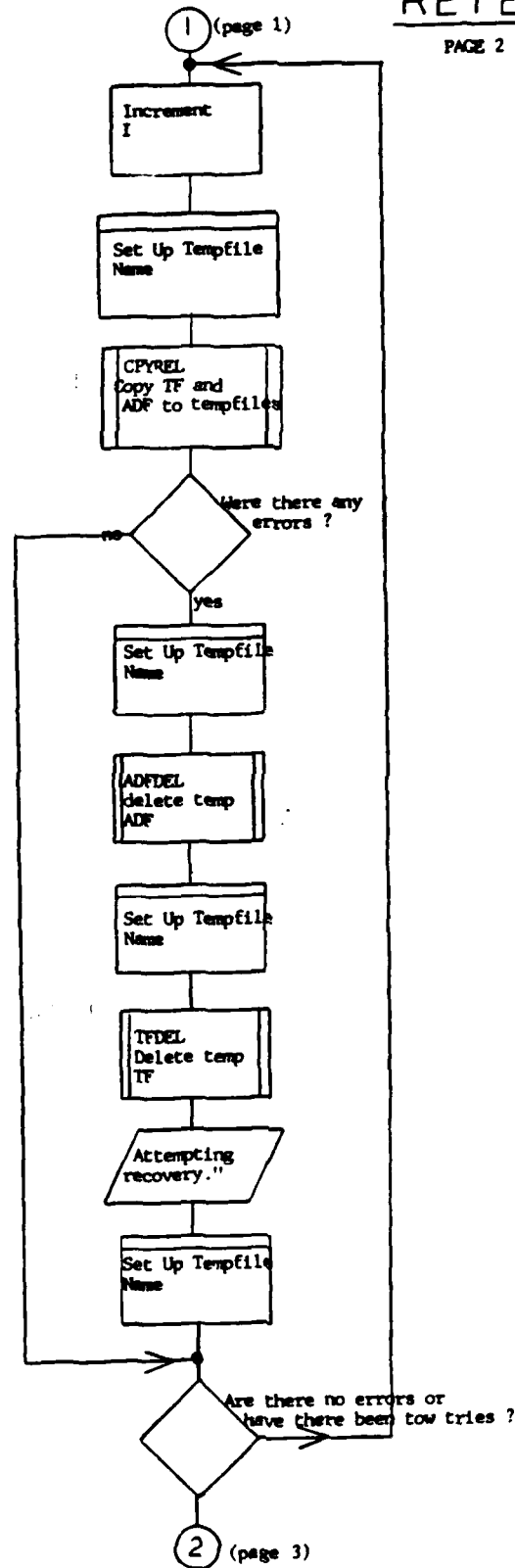
# RETEVD

PAGE 1



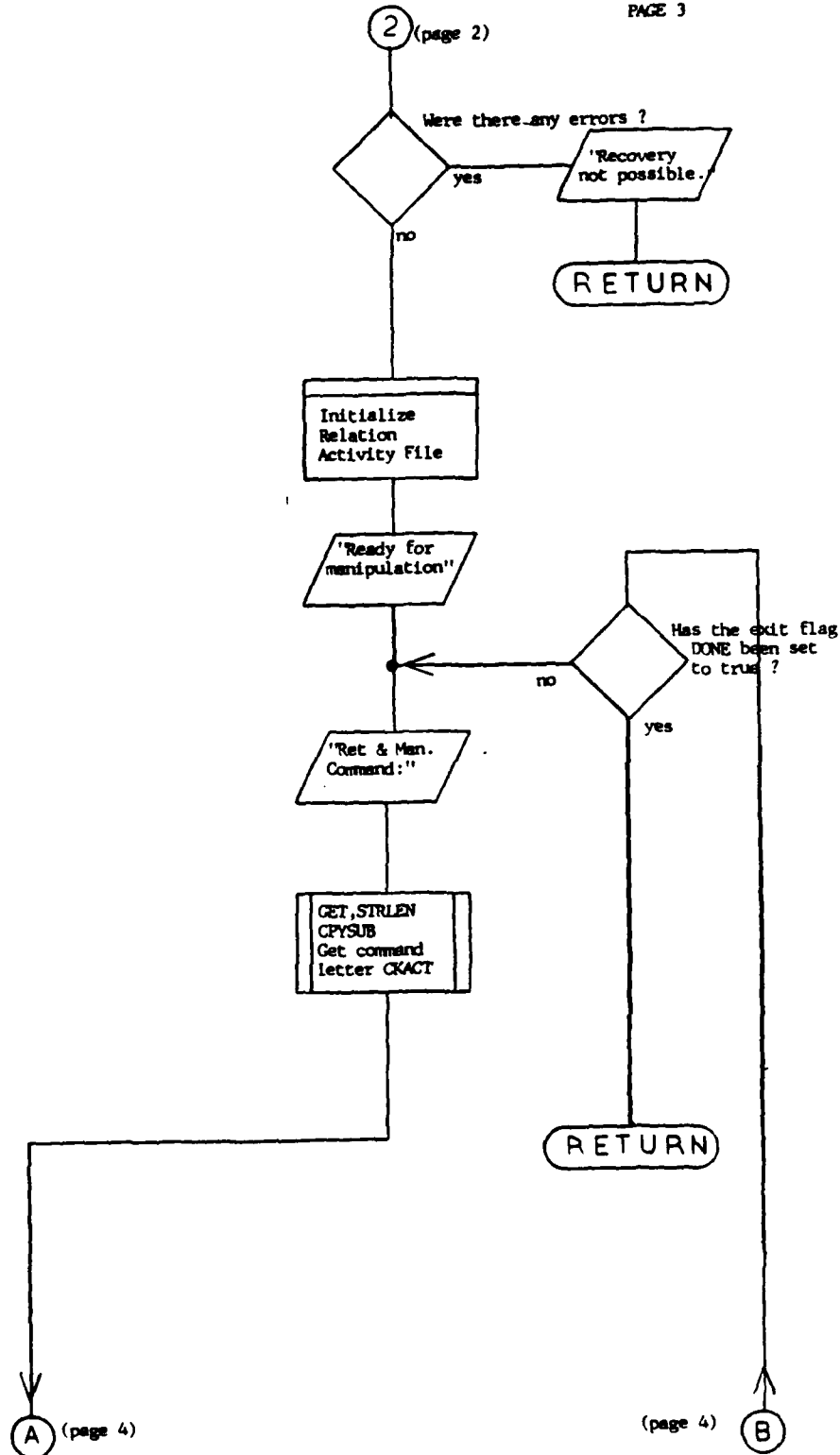
# RETEVD

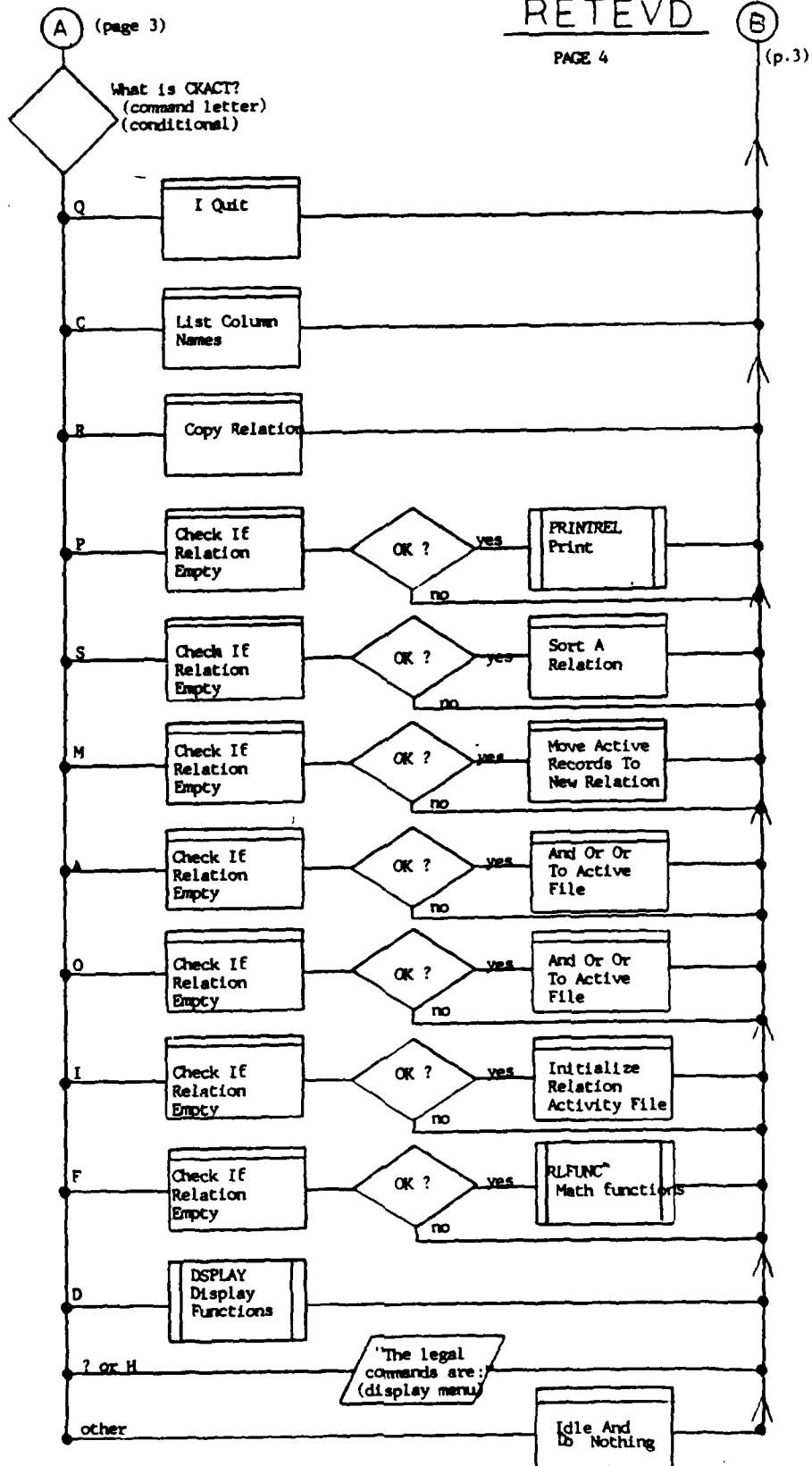
PAGE 2



# RETEVD

PAGE 3



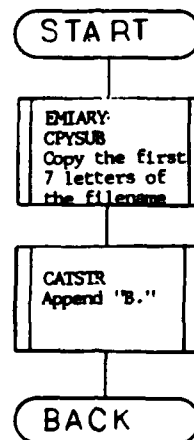


RETEVD

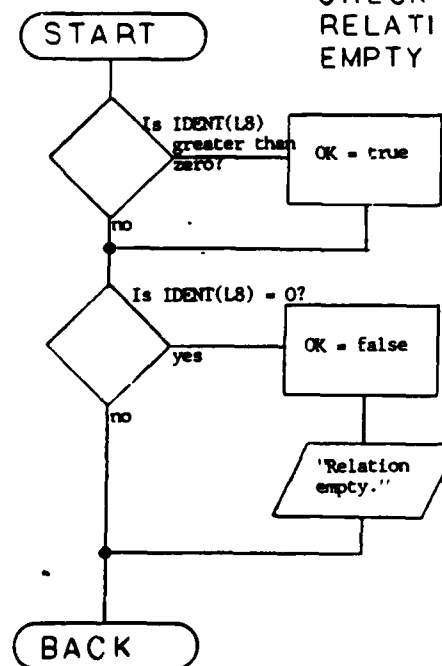
SET UP  
TEMPFILE

NAME

PAGE 5



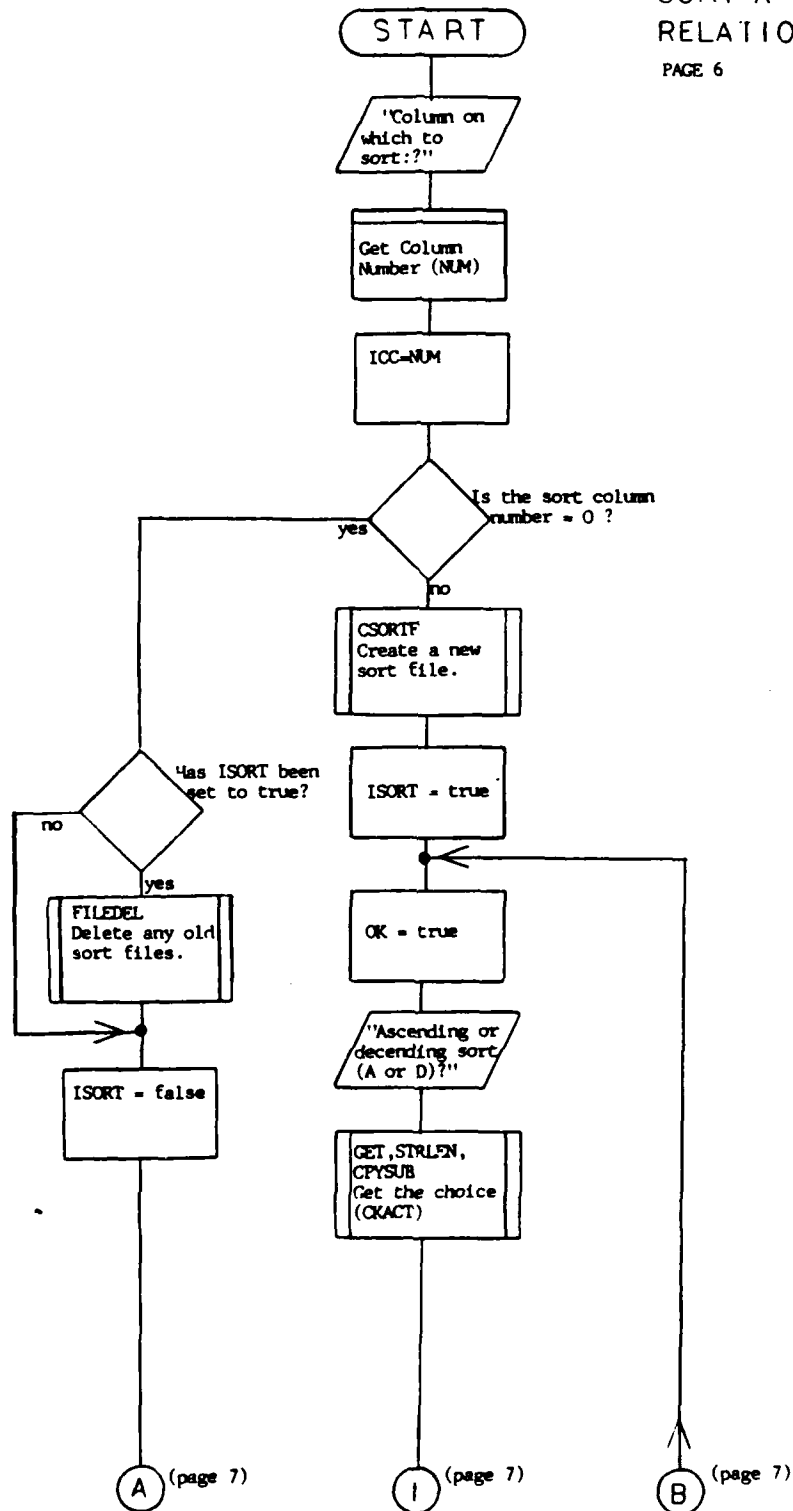
CHECK IF  
RELATION  
EMPTY

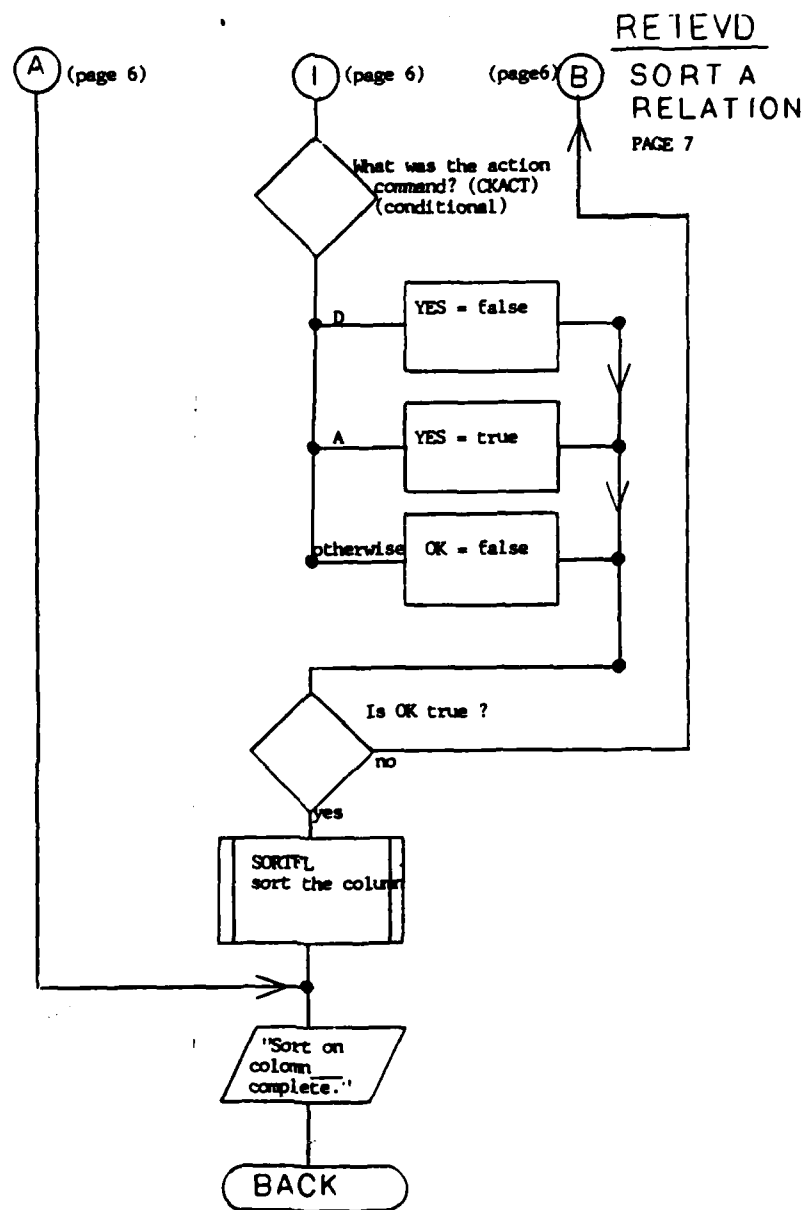


# RETEVD

## SORT A RELATION

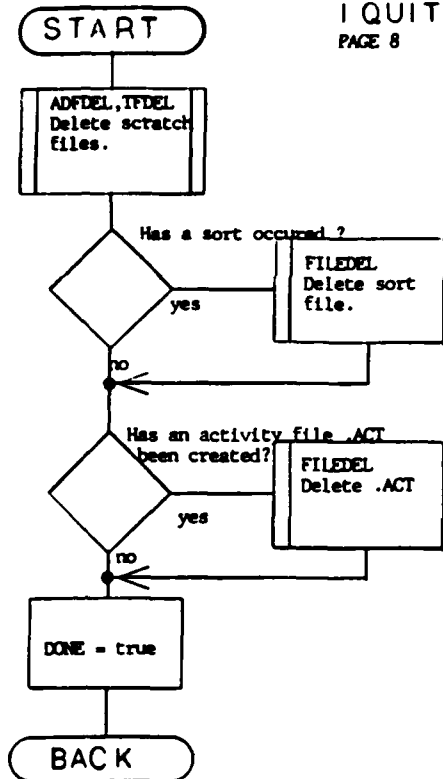
PAGE 6



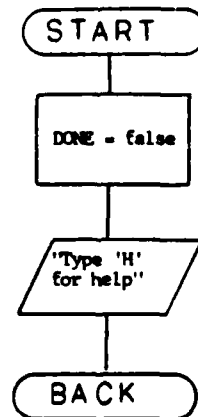


RETEVD

I QUIT  
PAGE 8



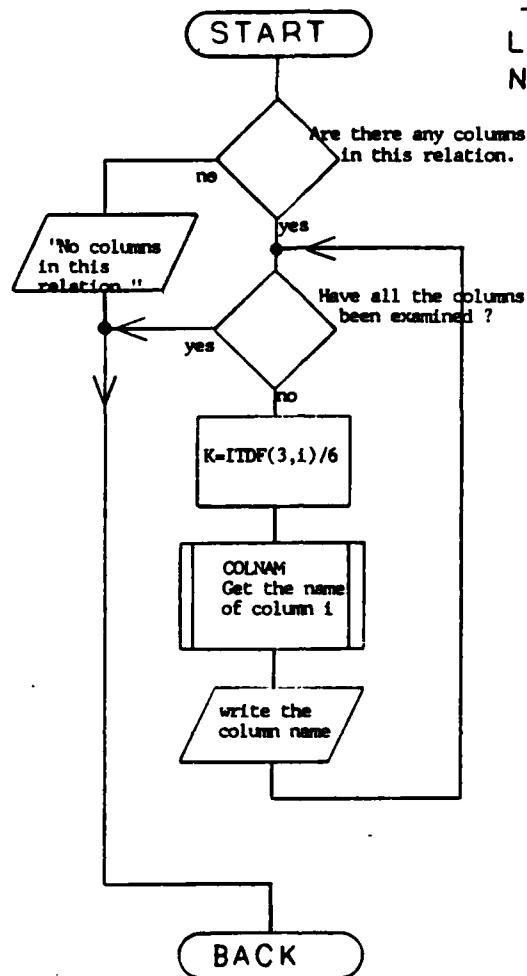
IDLE AND  
DO NOTHING





RETEVD  
LIST COLUMN  
NAMES

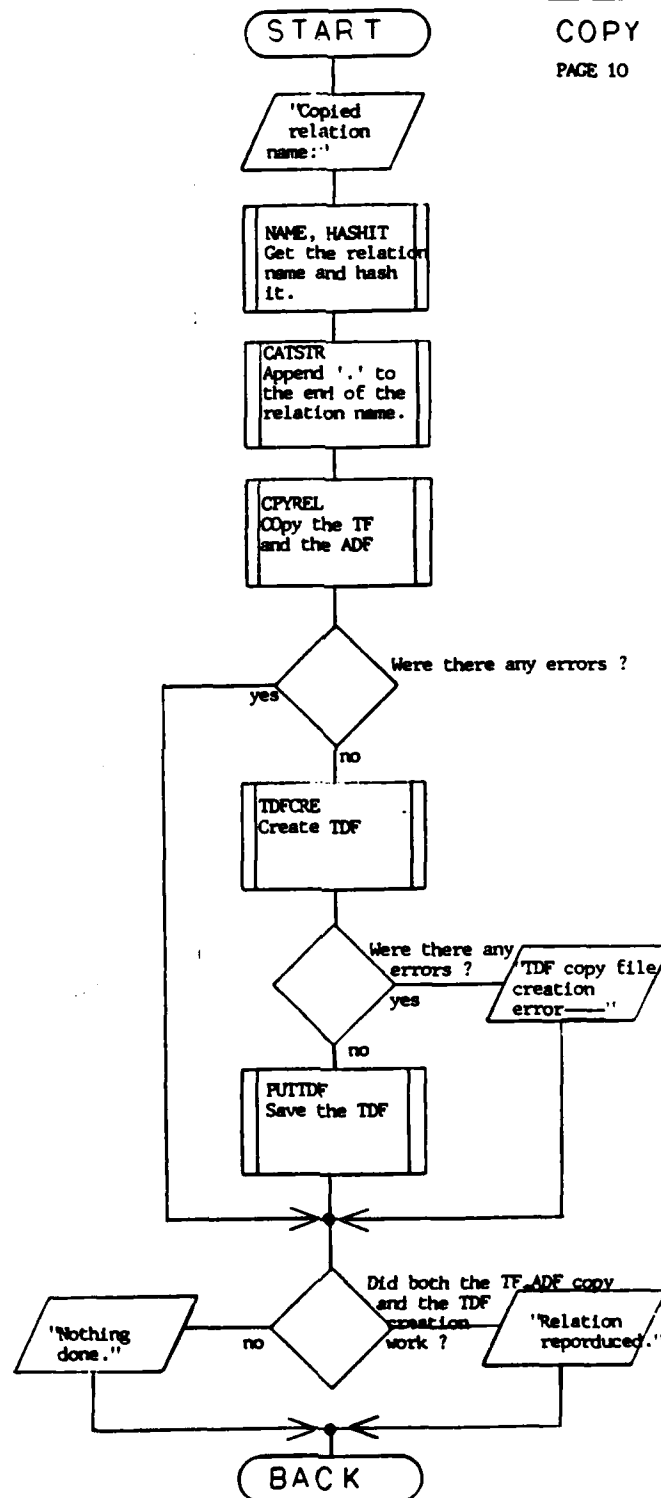
PAGE 9



# RETEVD

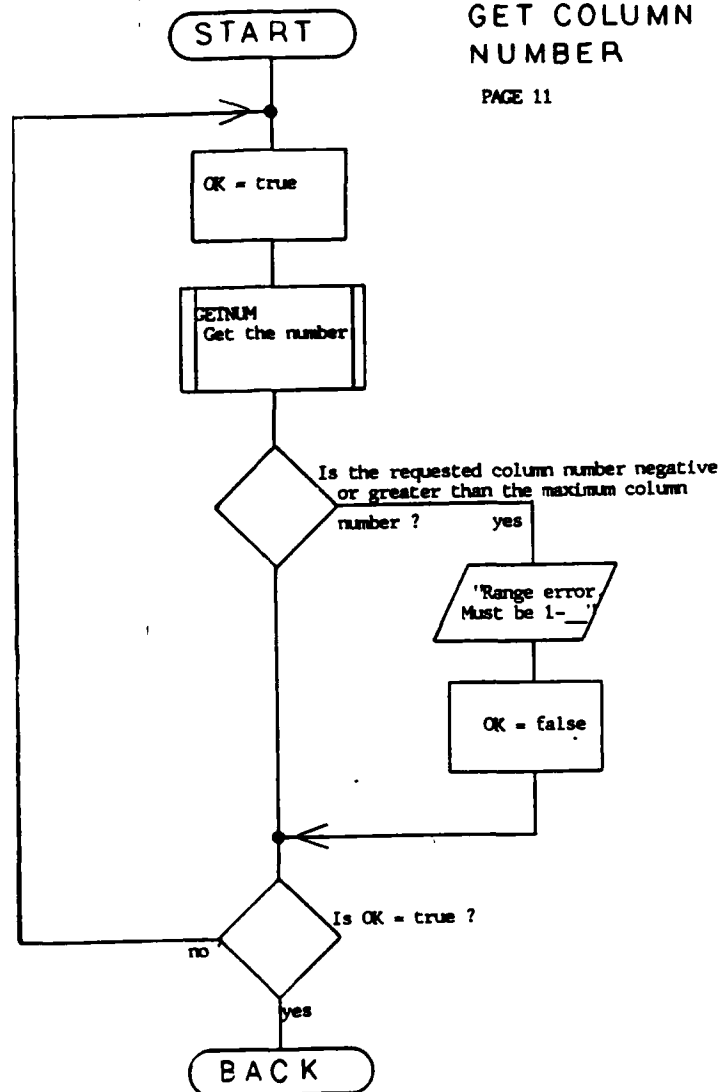
## COPY RELATION

PAGE 10



RETEVD  
GET COLUMN  
NUMBER

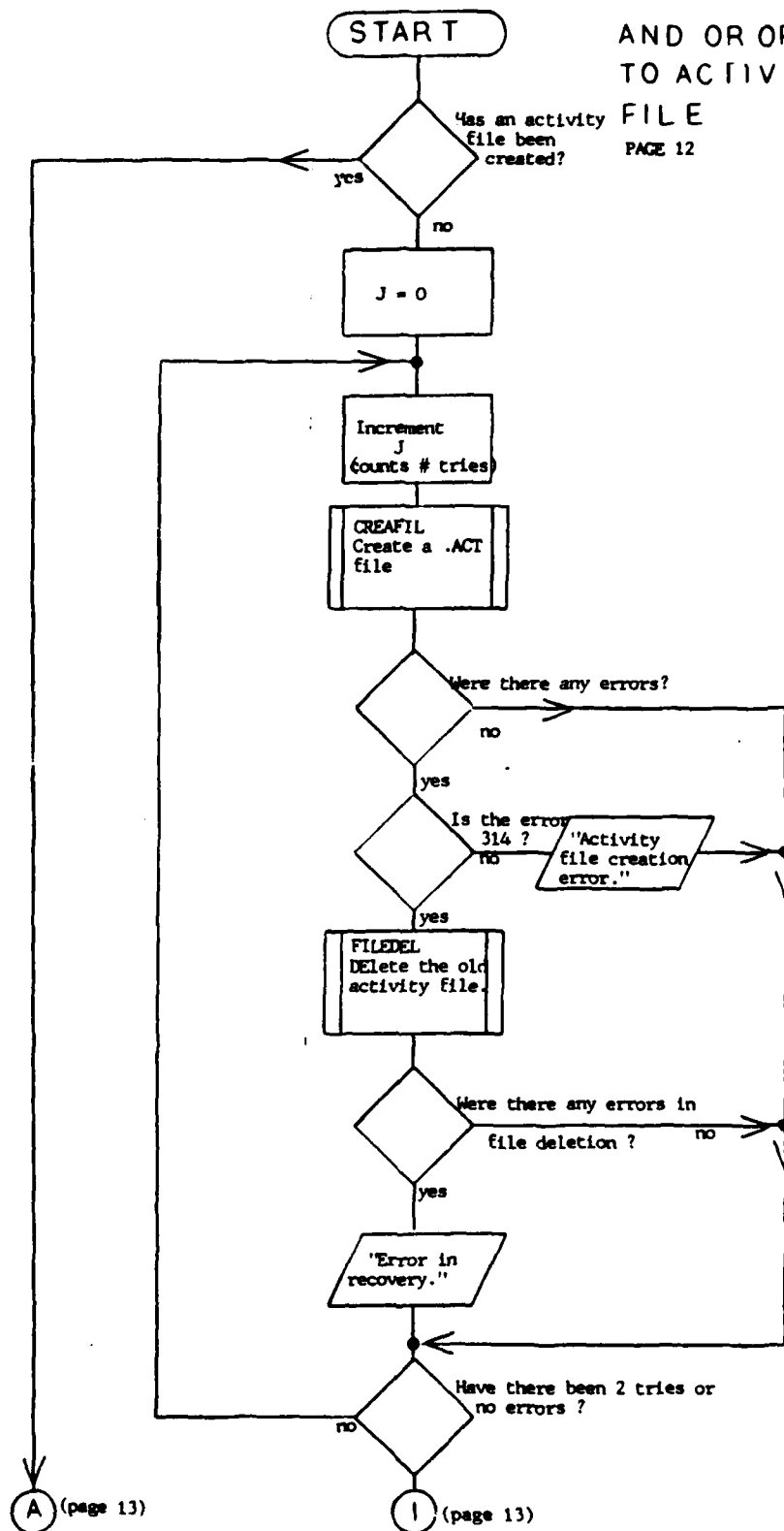
PAGE 11



RETEVD

AND OR OR  
TO ACTIVE  
FILE

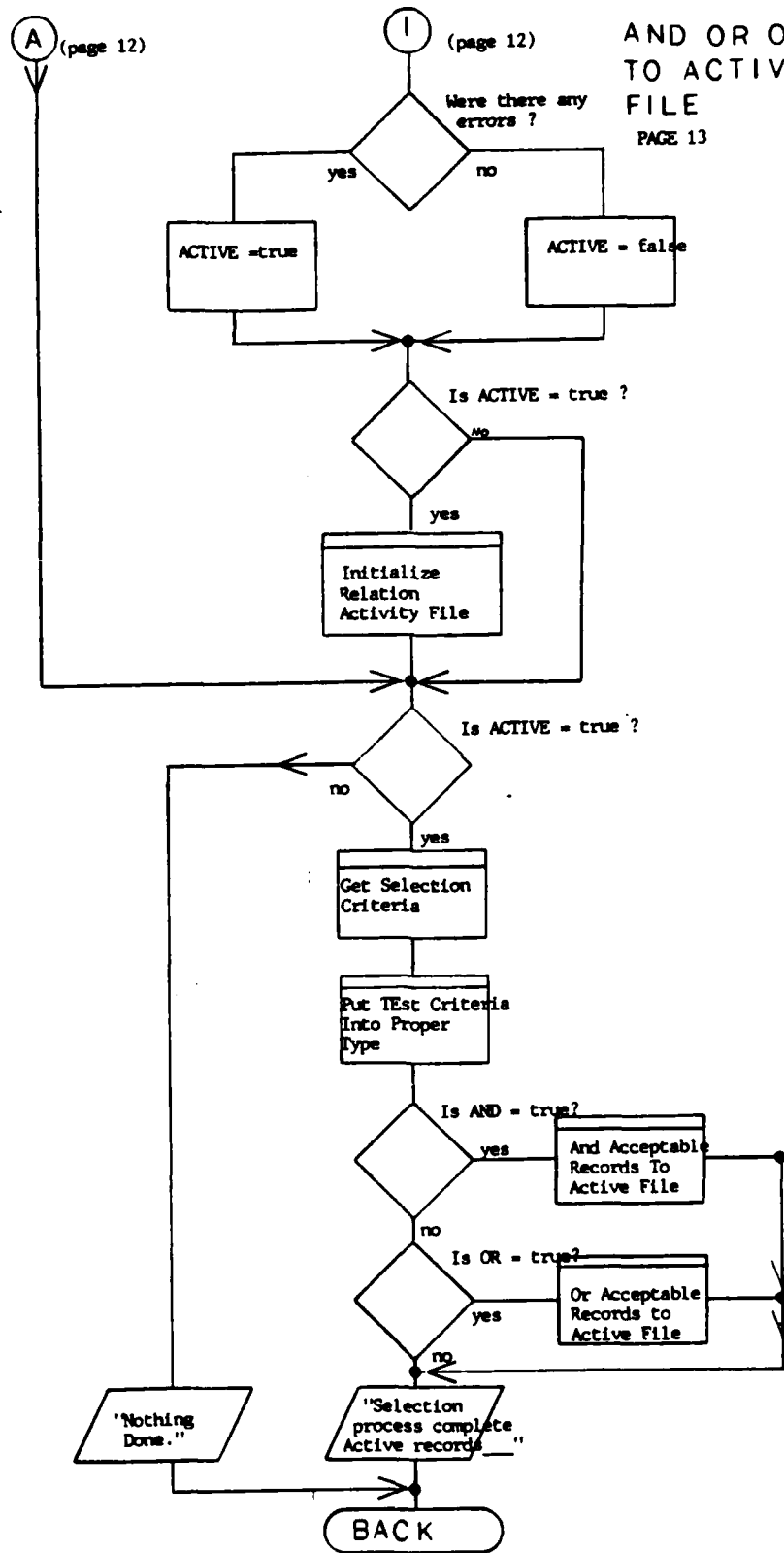
PAGE 12



# RETEVD

AND OR OR  
TO ACTIVE  
FILE

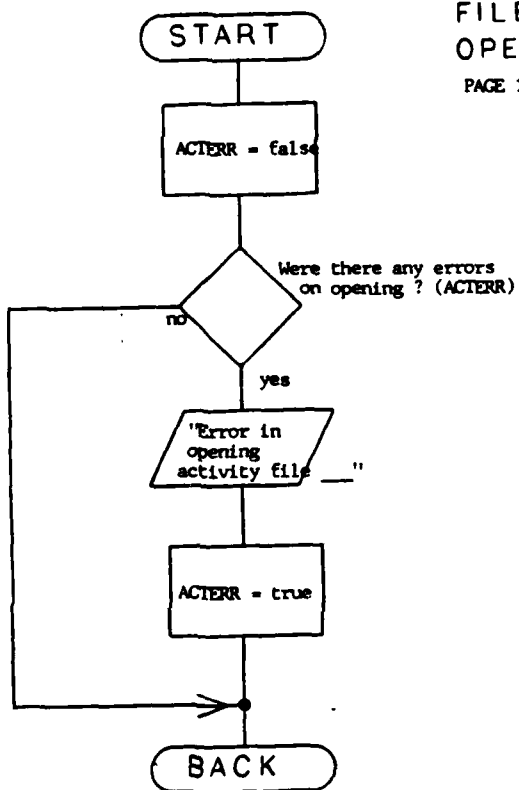
PAGE 13



RETEVD

CHECK  
ACTIVITY  
FILE  
OPENING

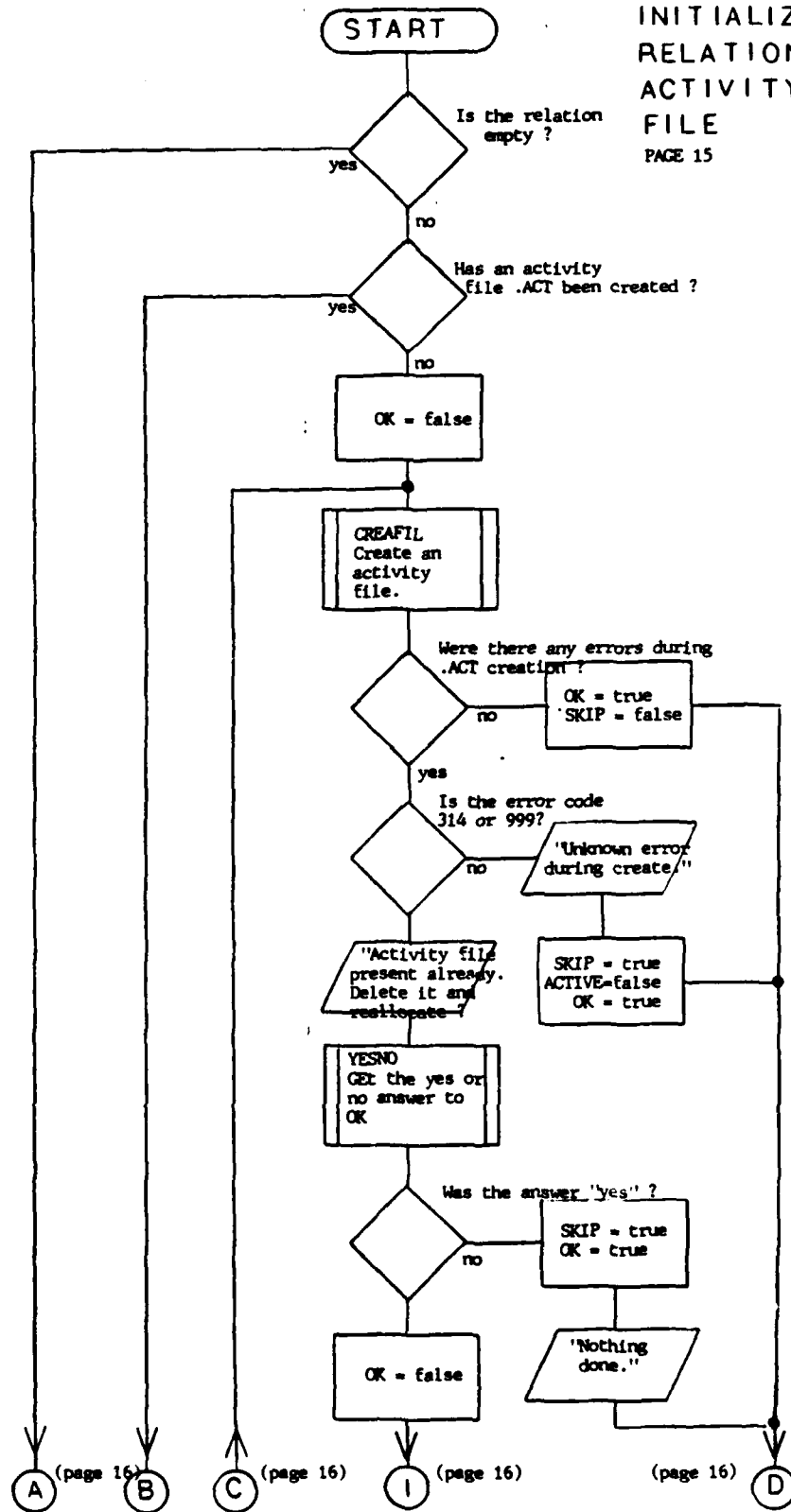
PAGE 14

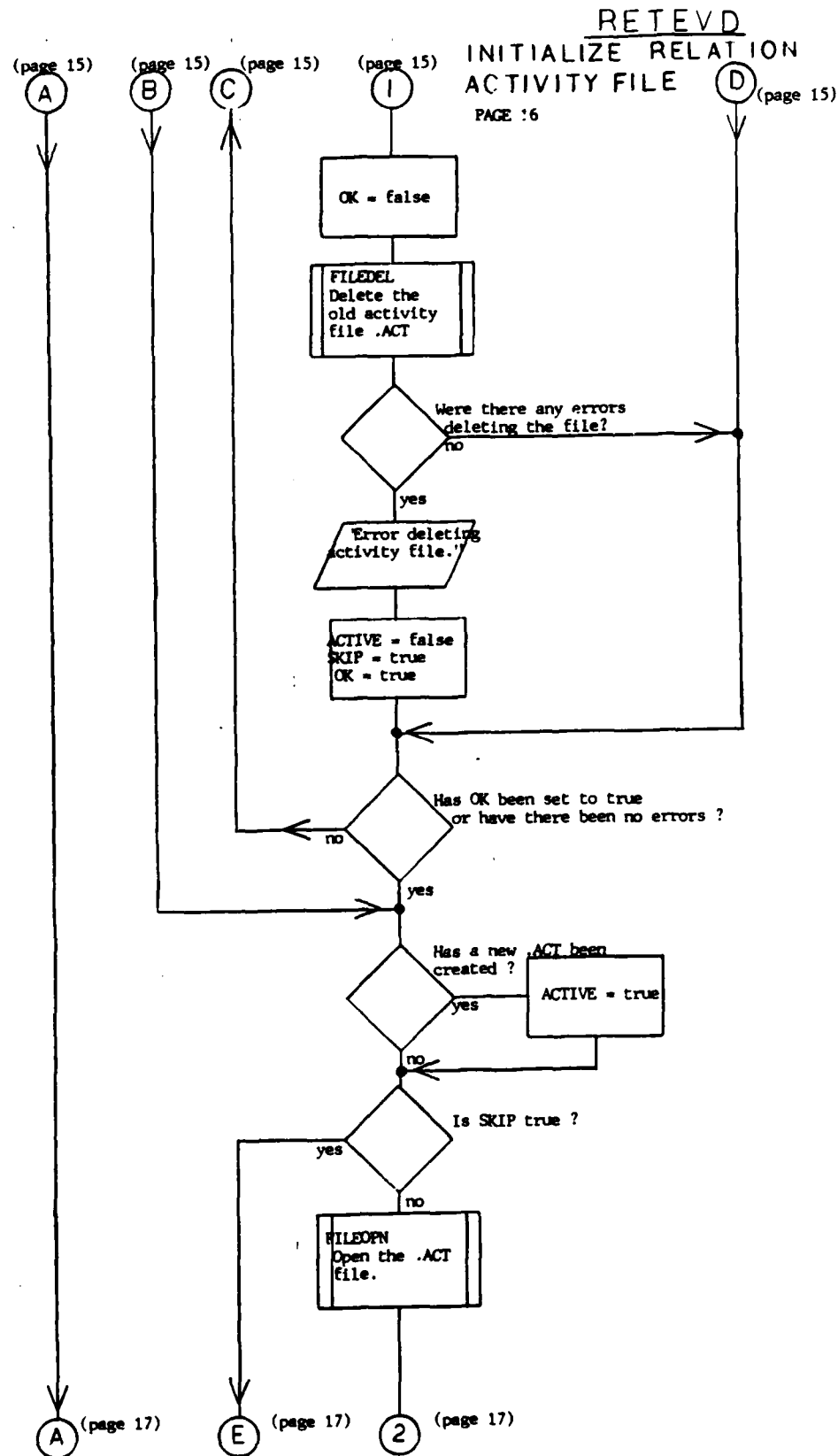


# RETEVD

INITIALIZE  
RELATION  
ACTIVITY  
FILE

PAGE 15

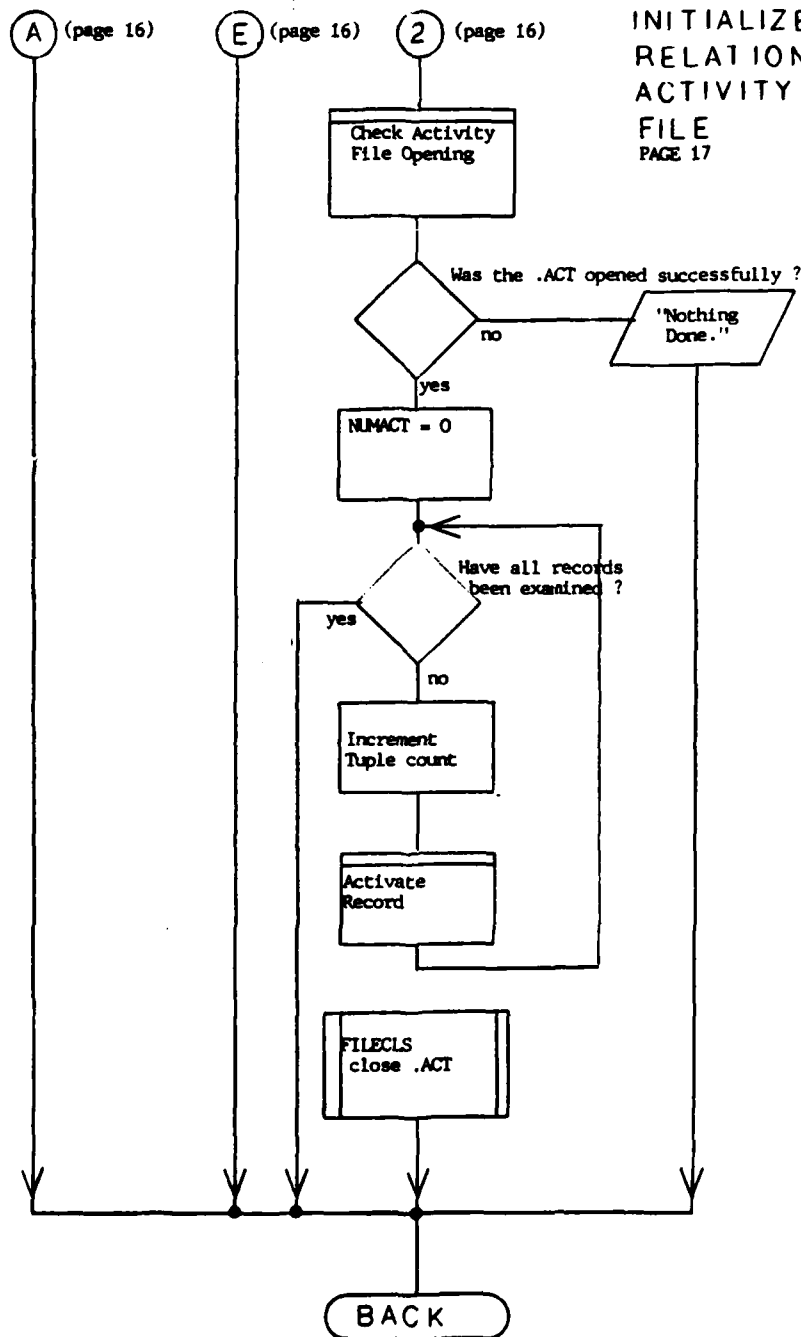






RETEVD

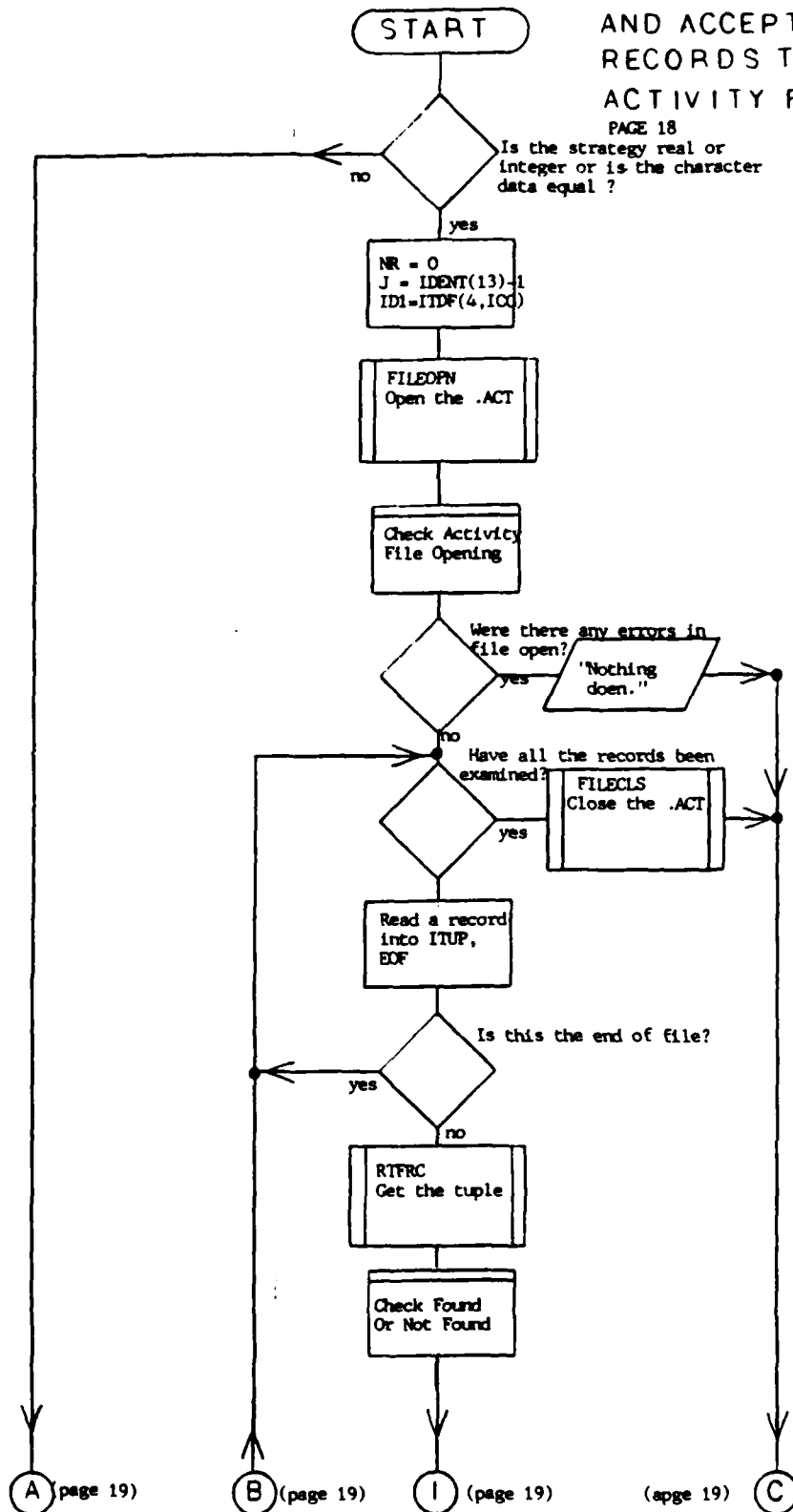
INITIALIZE  
RELATION  
ACTIVITY  
FILE  
PAGE 17

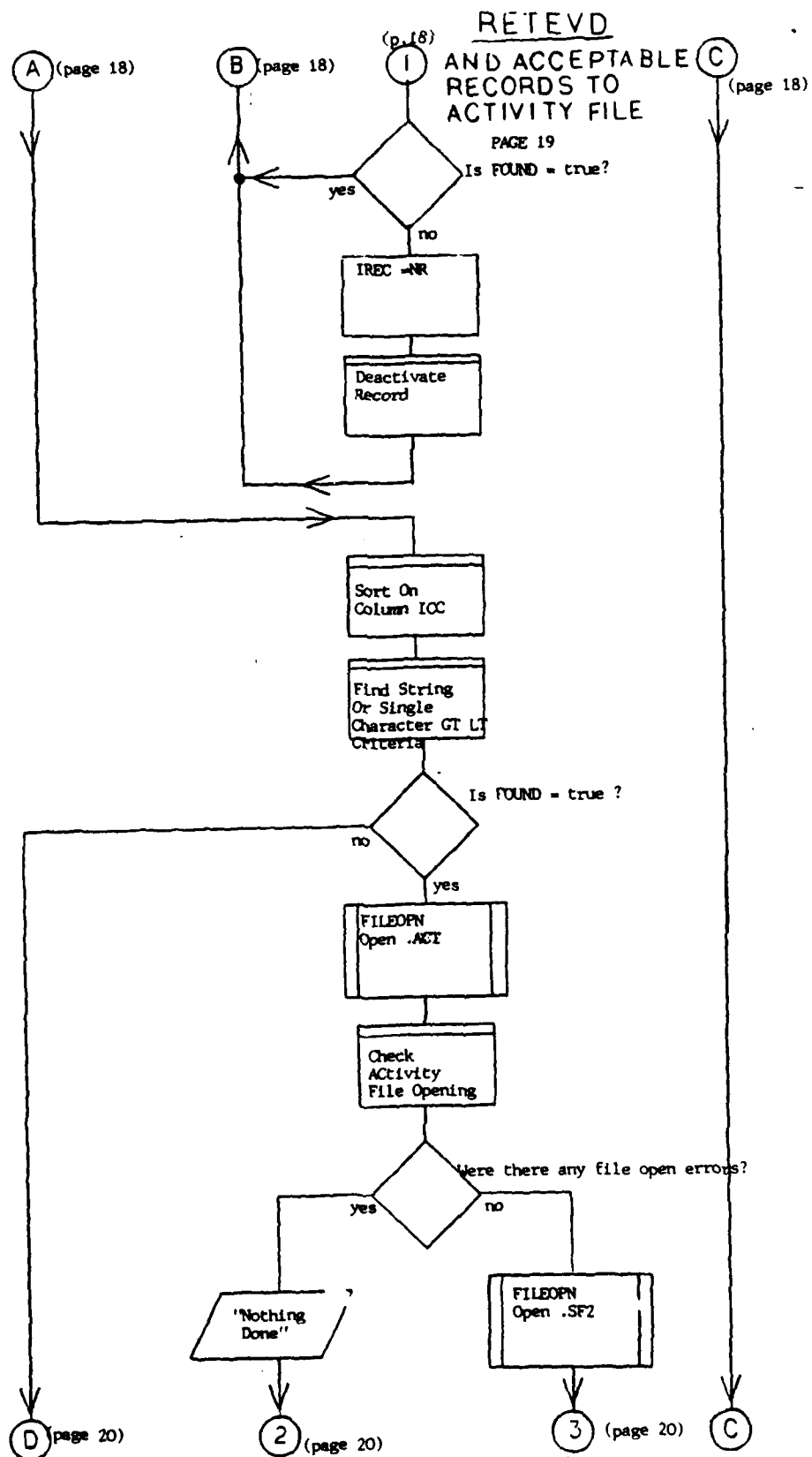


# RETEVD

## AND ACCEPTABLE RECORDS TO ACTIVITY FILE

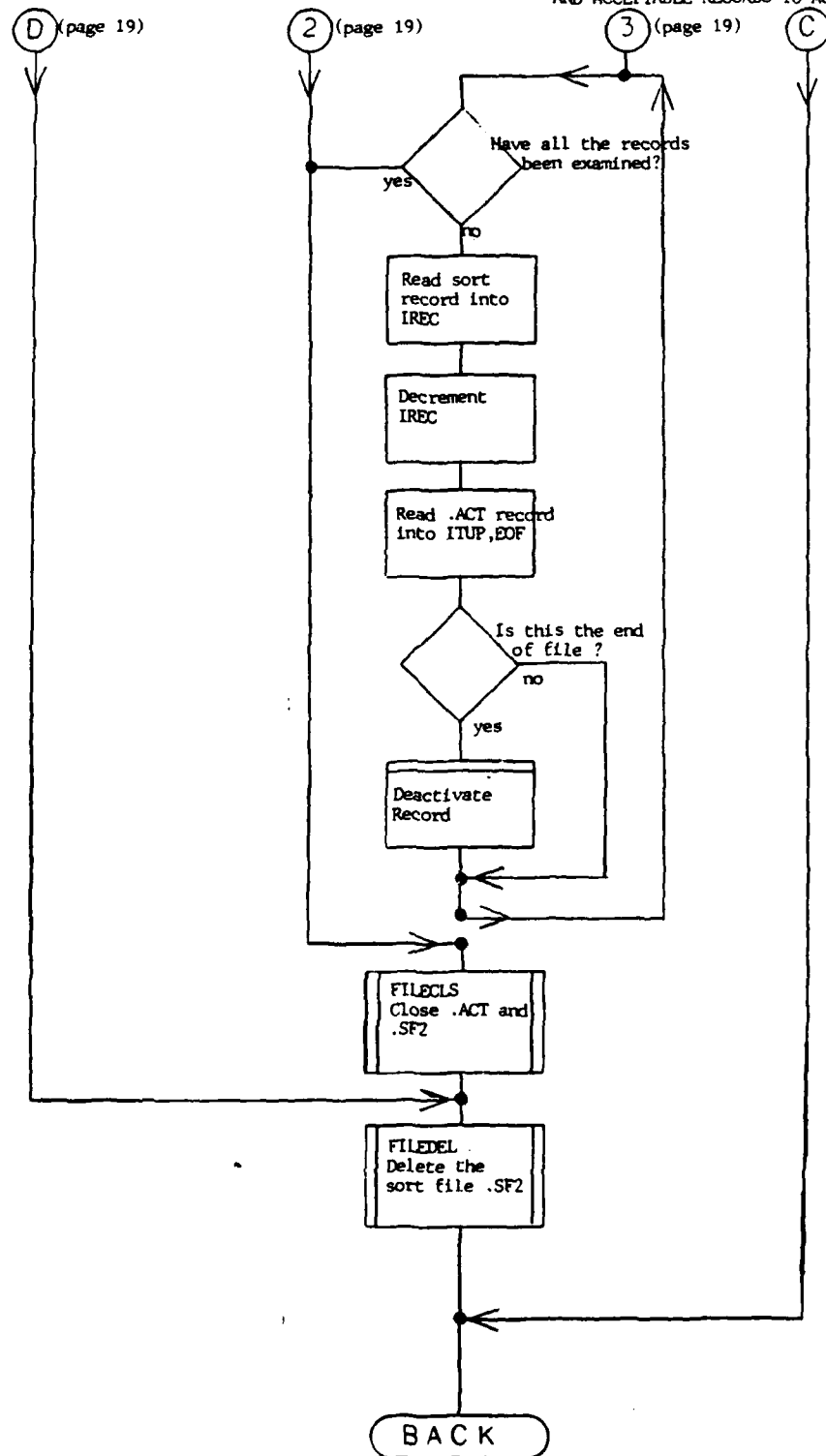
PAGE 18



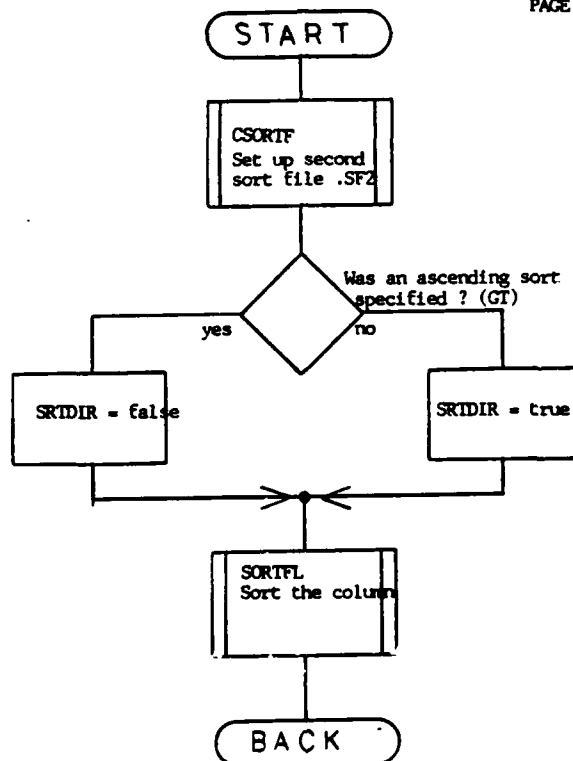


RETEVD  
AND ACCEPTABLE RECORDS TO ACTIVITY  
FILE

PAGE  
20



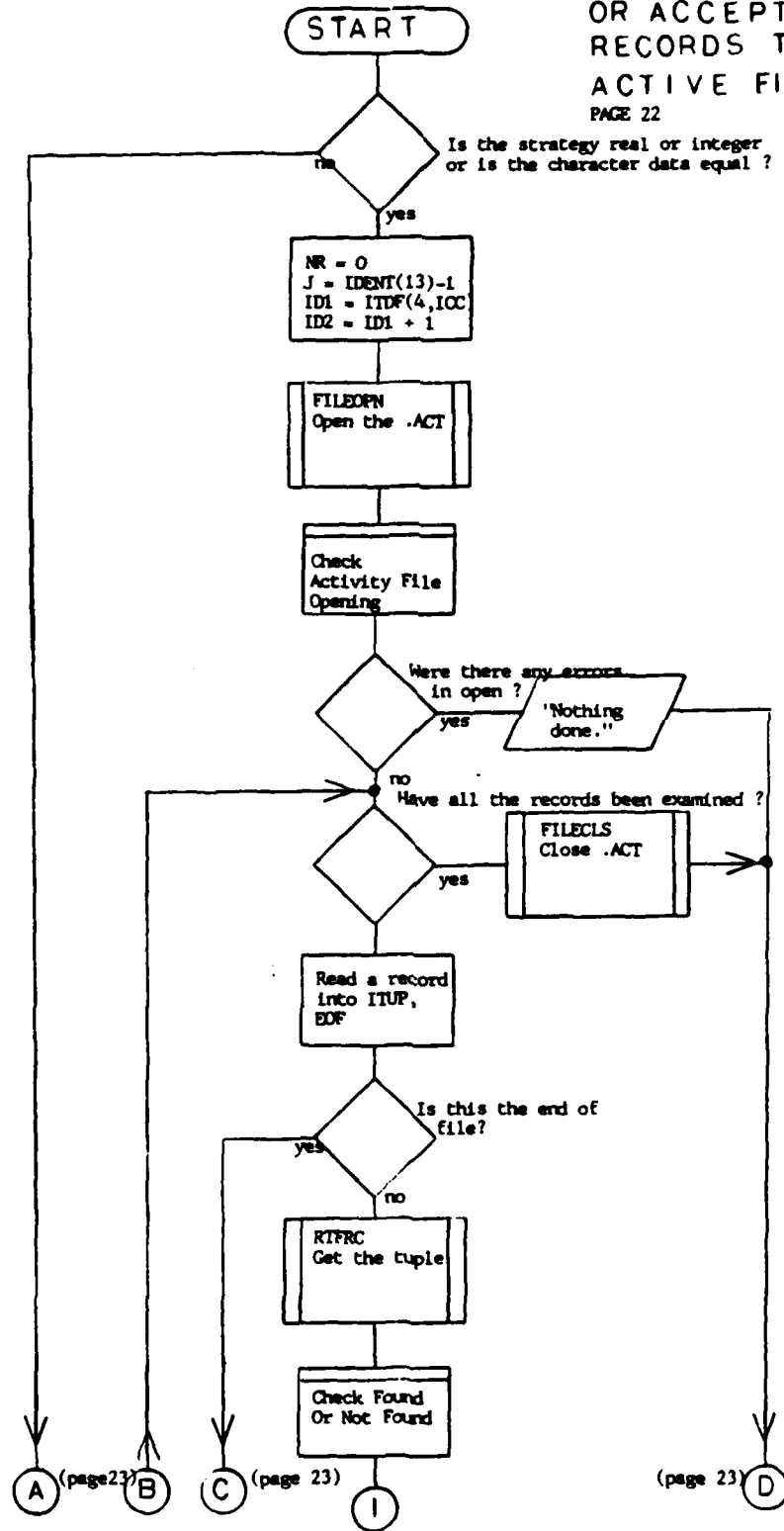
RELEV  
SORT ON  
COLUMN ICC  
PAGE 21

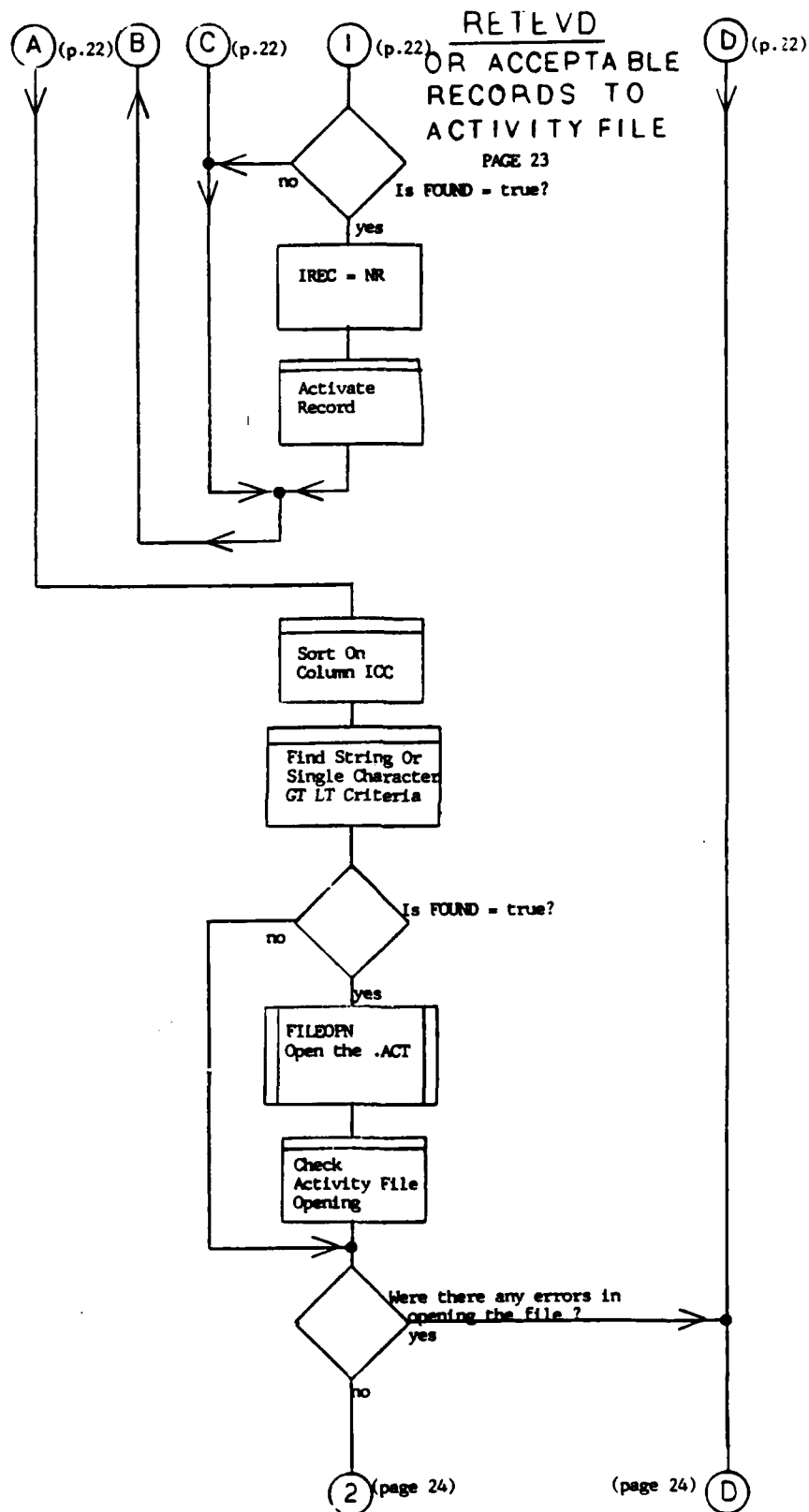


# RETEVD

OR ACCEPTABLE  
RECORDS TO  
ACTIVE FILE

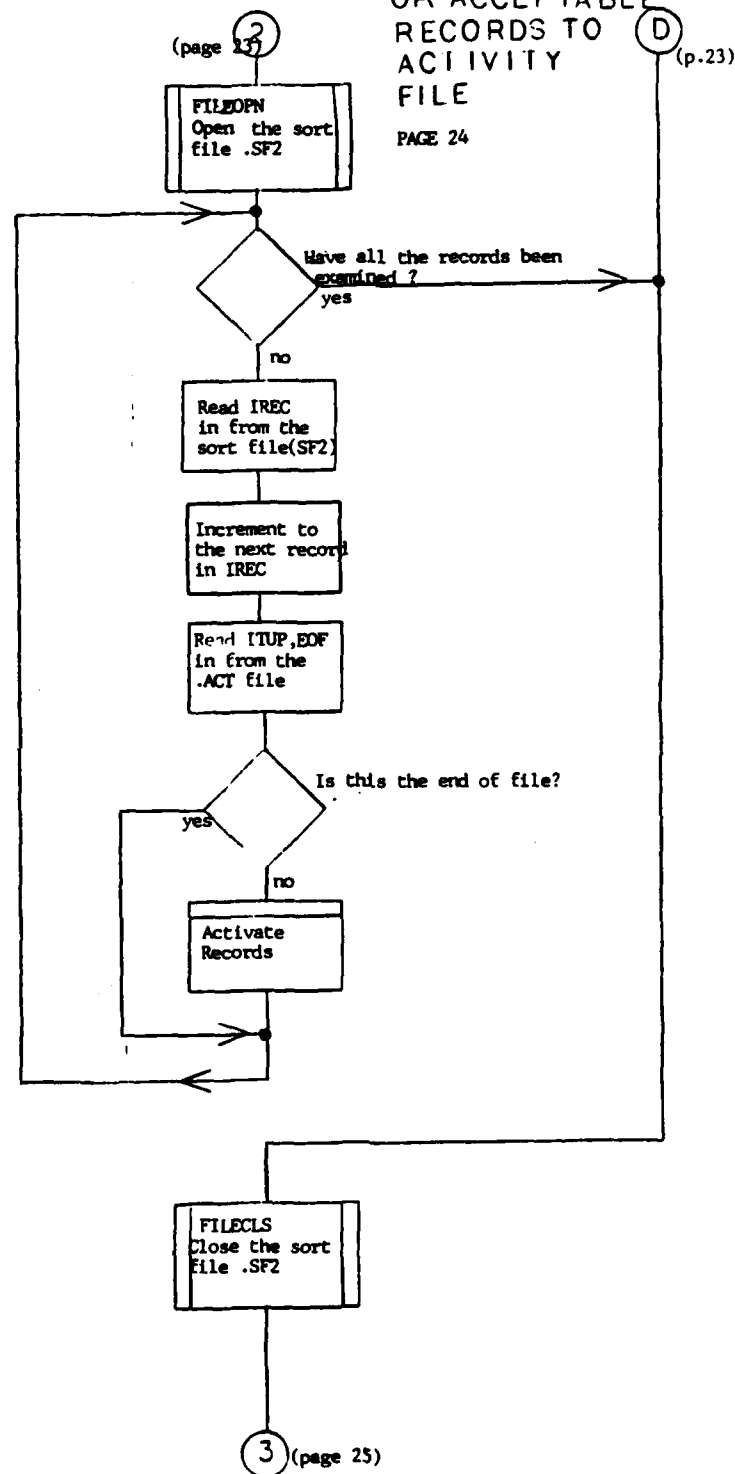
PAGE 22





RELEV  
OR ACCEPTABLE  
RECORDS TO  
ACTIVITY  
FILE

PAGE 24

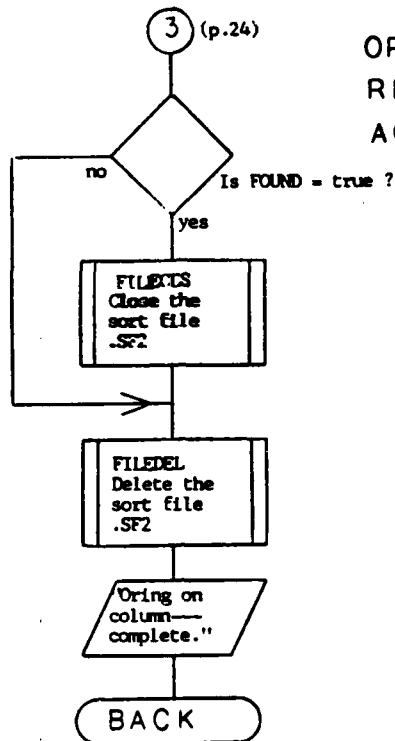




RE TEVD

OR ACCEPTABLE  
RECORDS TO  
ACTIVITY FILE

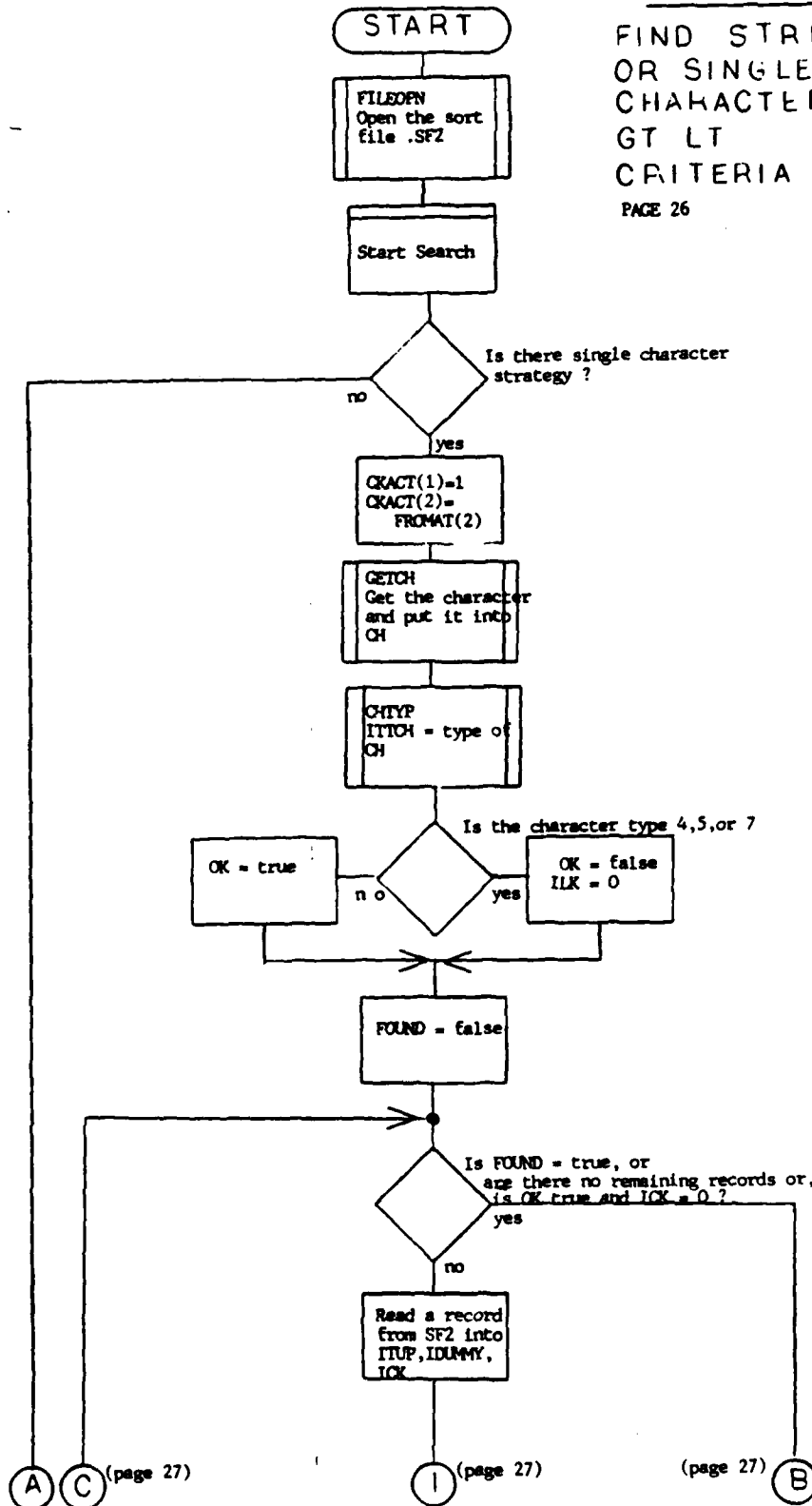
PAGE 25



# RETEVD

FIND STRING  
OR SINGLE  
CHARACTER  
GT LT  
CRITERIA

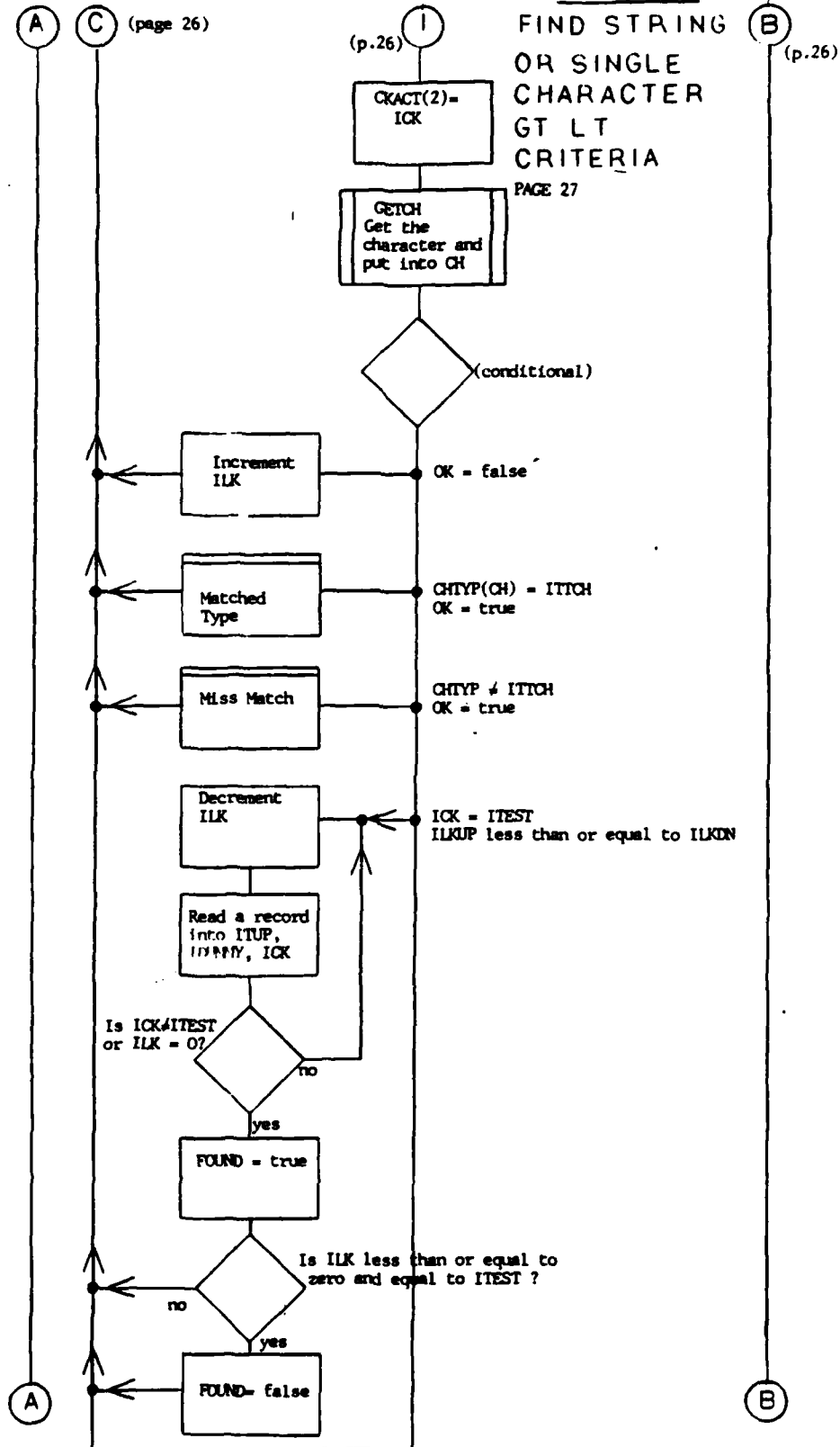
PAGE 26



# RETEVD

FIND STRING  
OR SINGLE  
CHARACTER  
GT LT  
CRITERIA

PAGE 27

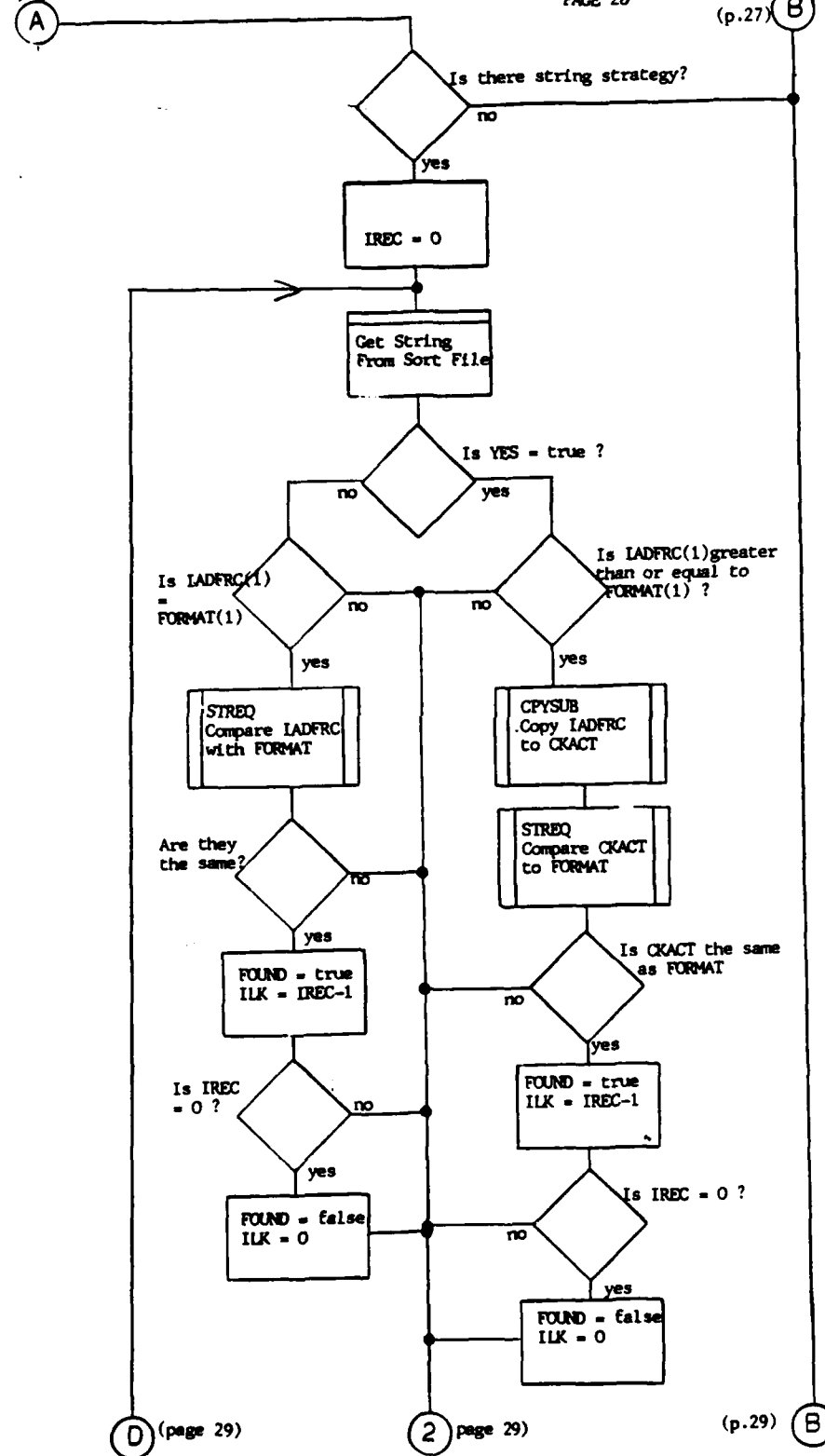


# RELEV

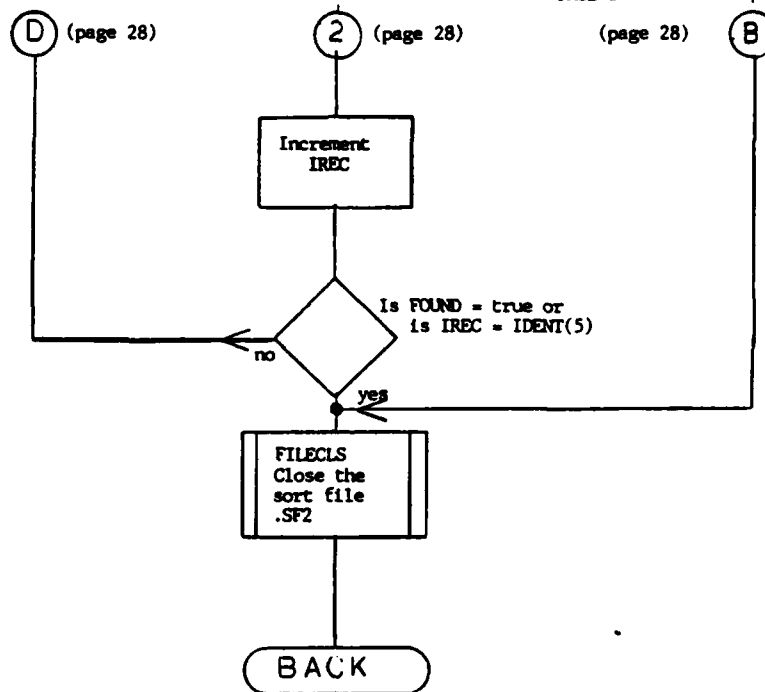
FIND STRING OR SINGLE CHARACTER GT LT CRITERIA  
PAGE 28

(page 27)

(p.27)



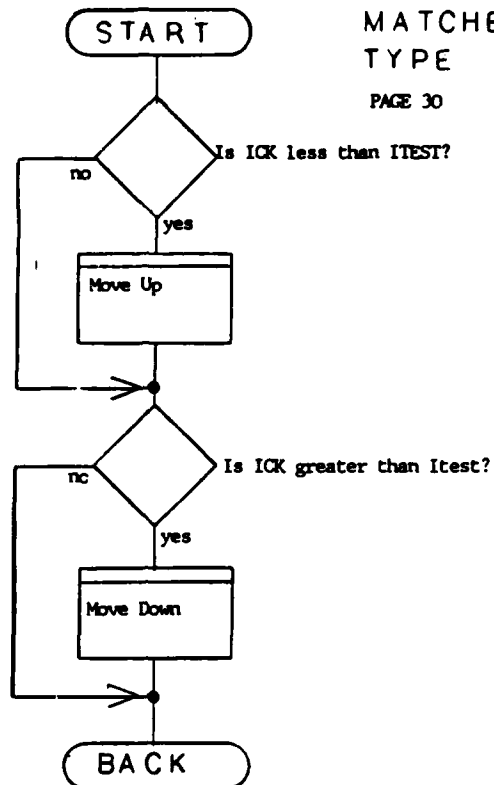
RETEVD  
PAGE 29



# RETEVD

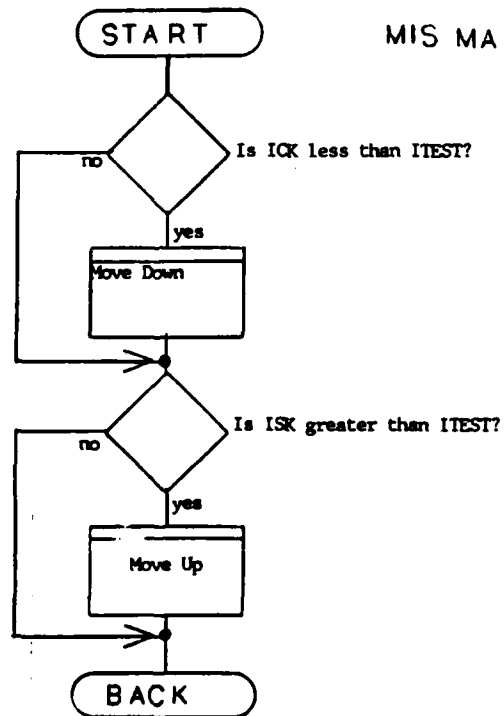
MATCHED  
TYPE

PAGE 30



---

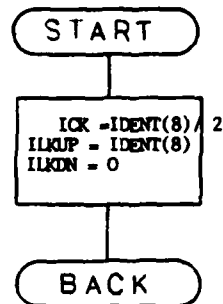
MIS MATCH



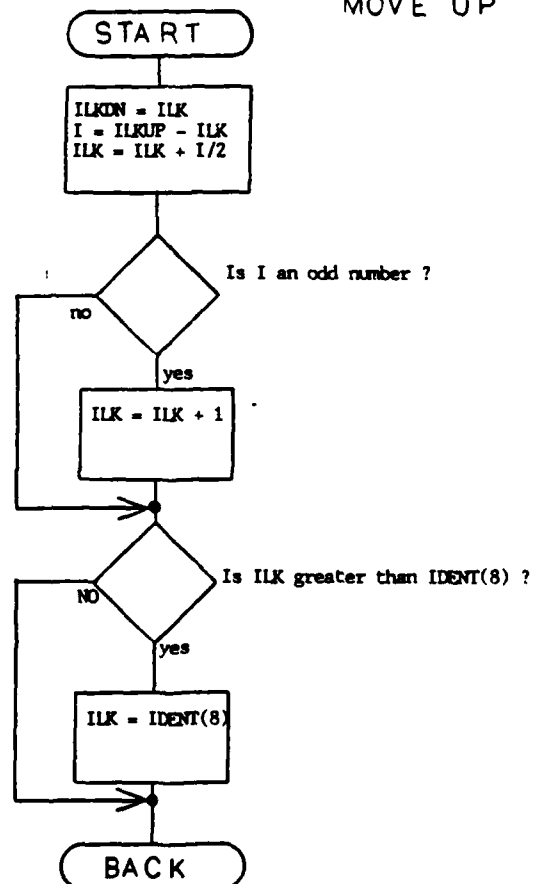
RETEVD

START SEARCH

PAGE 31



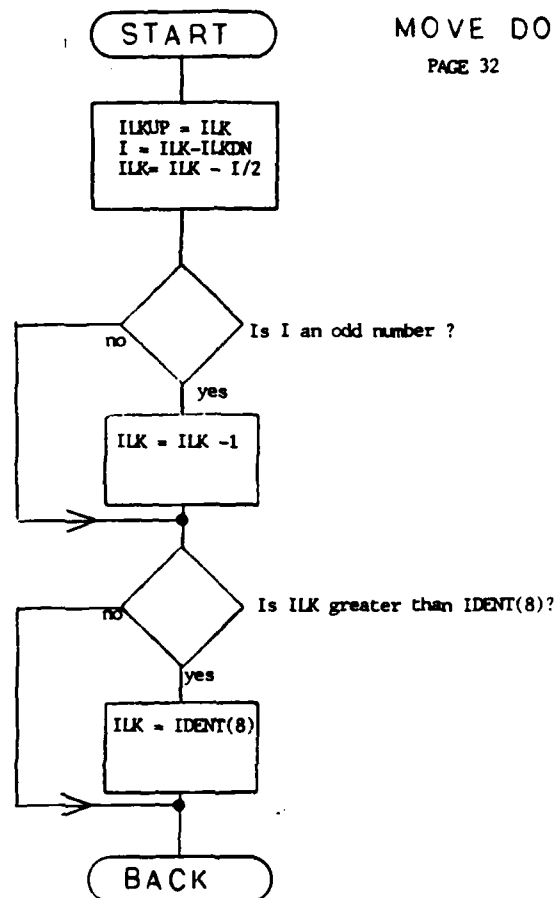
MOVE UP



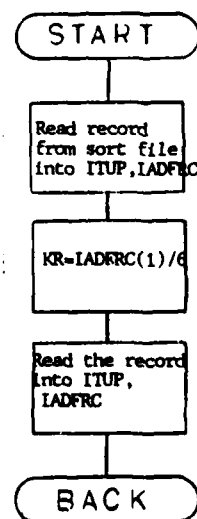
# RETEVD

MOVE DOWN

PAGE 32

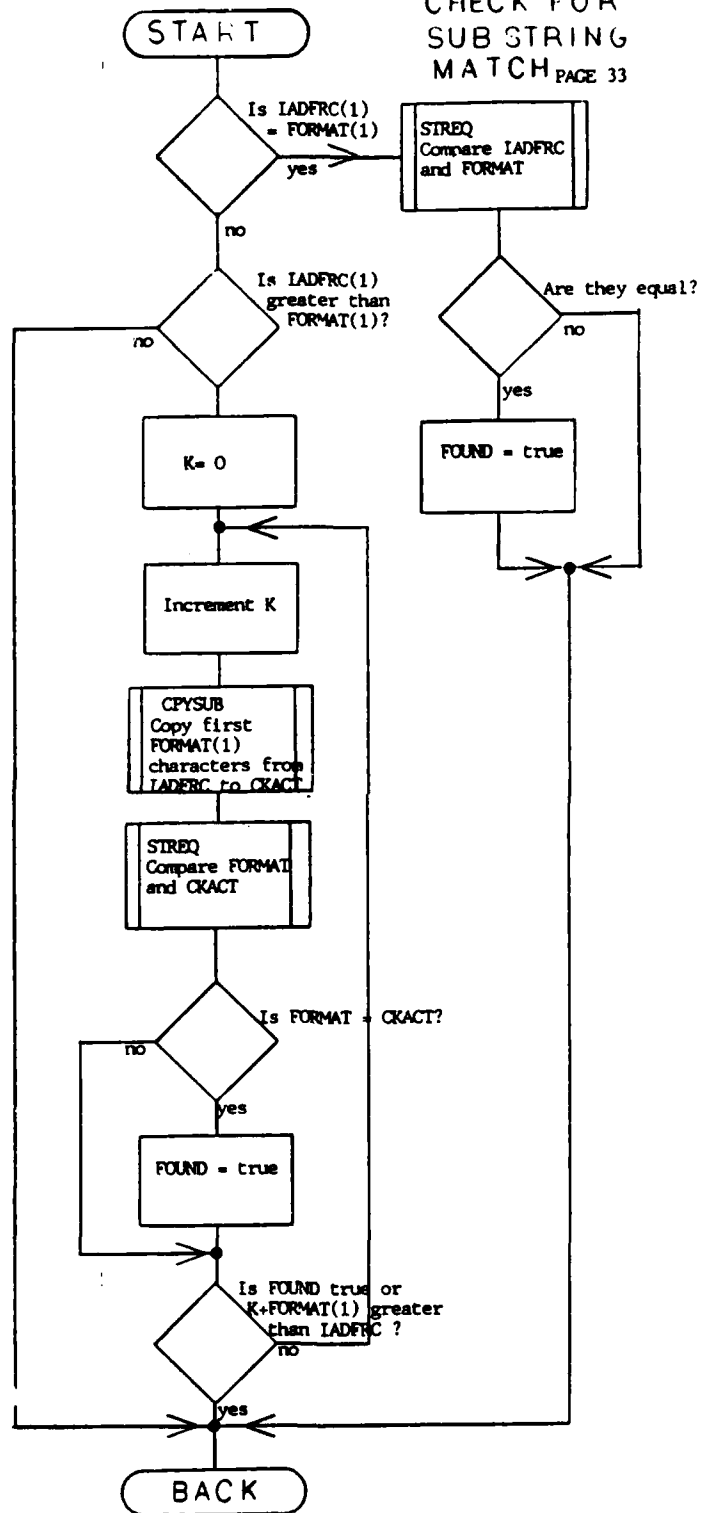


GET STRING  
FROM SORT  
FILE





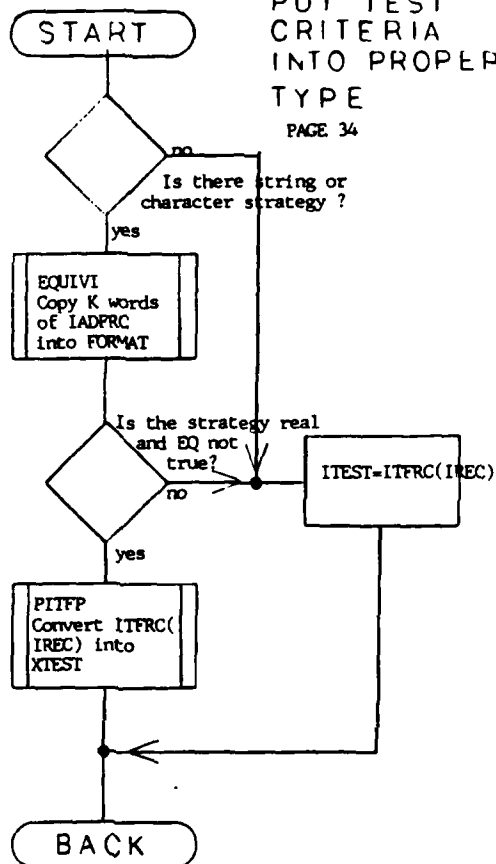
RETEVD  
CHECK FOR  
SUB STRING  
MATCH PAGE 33



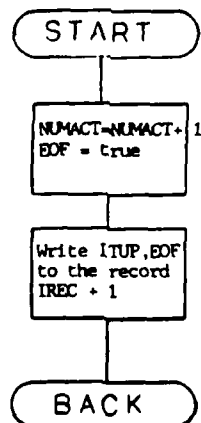
# RETEVD

PUT TEST  
CRITERIA  
INTO PROPER  
TYPE

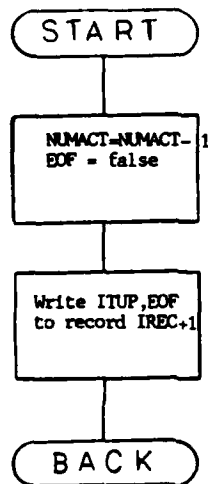
PAGE 34



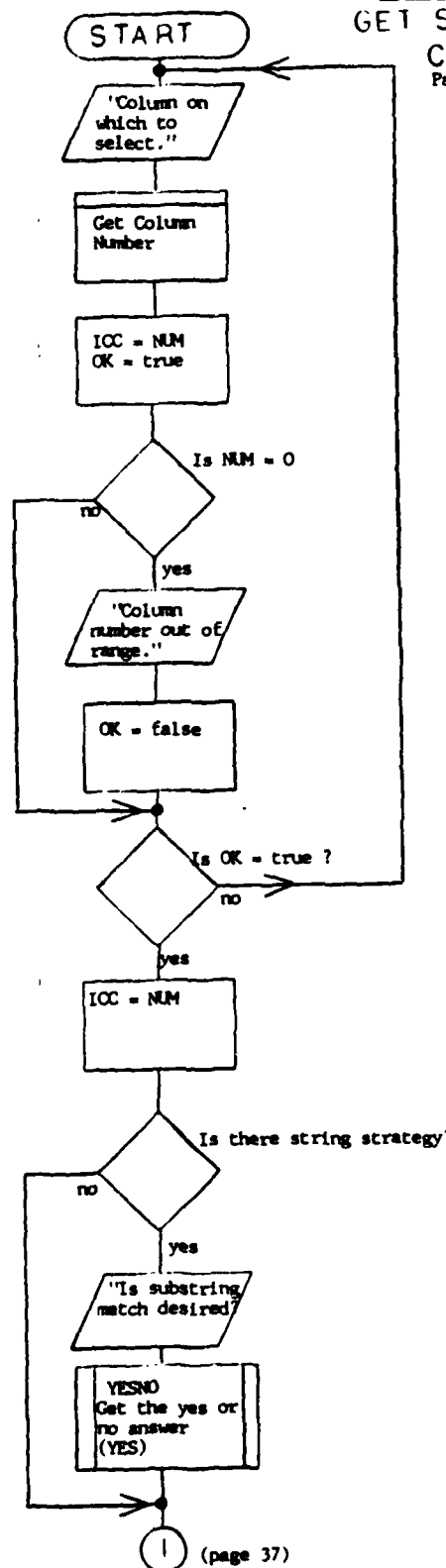
ACTIVATE  
RECORD



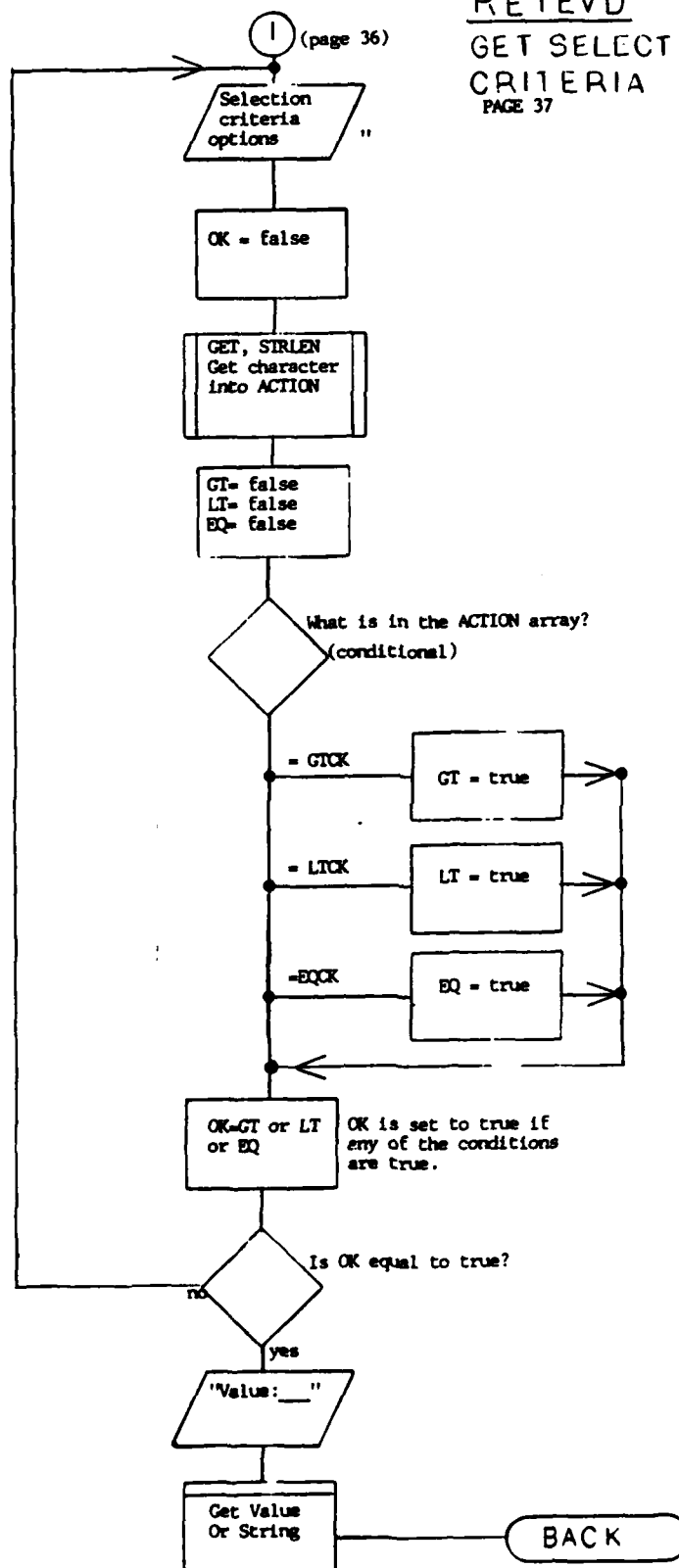
RETEVD  
DEACTIVATE  
RECORD  
PAGE 35



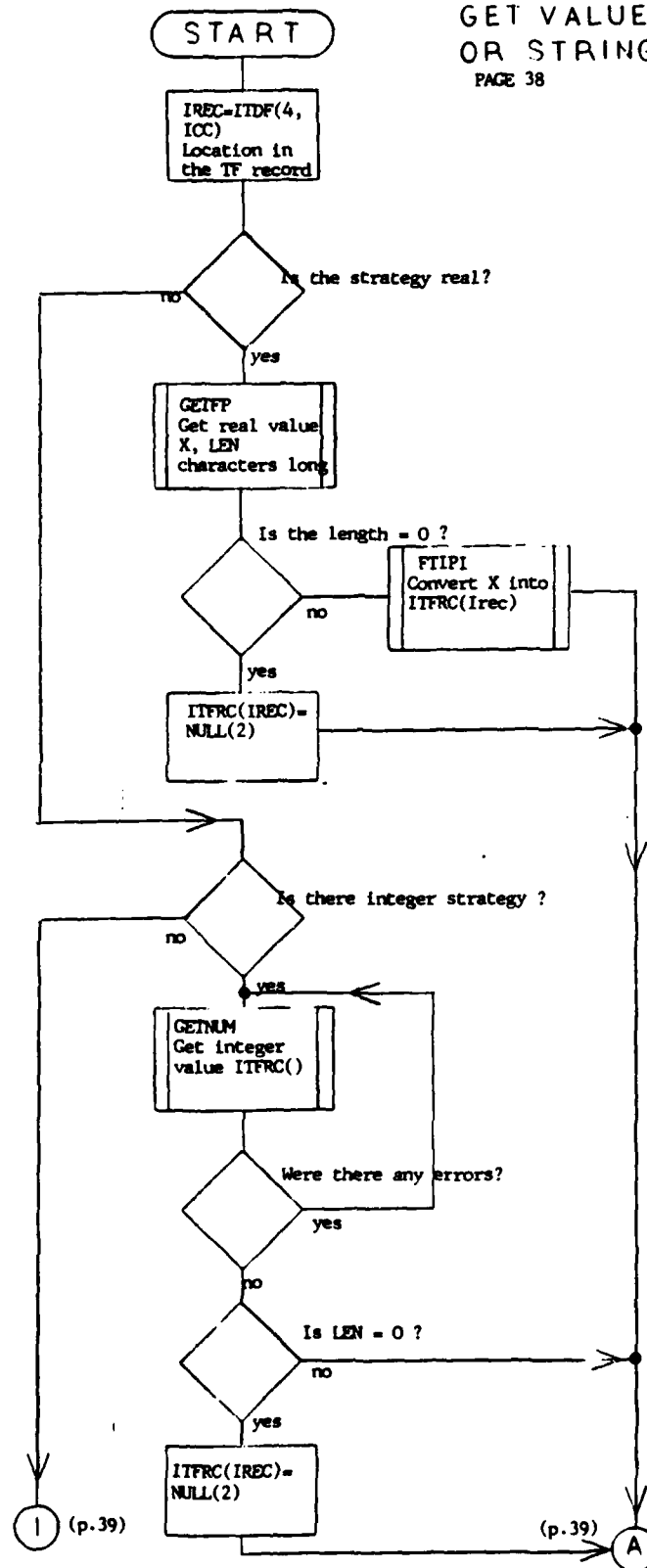
RETEVD  
GET SELECTION.  
CRITERIA  
Page 36



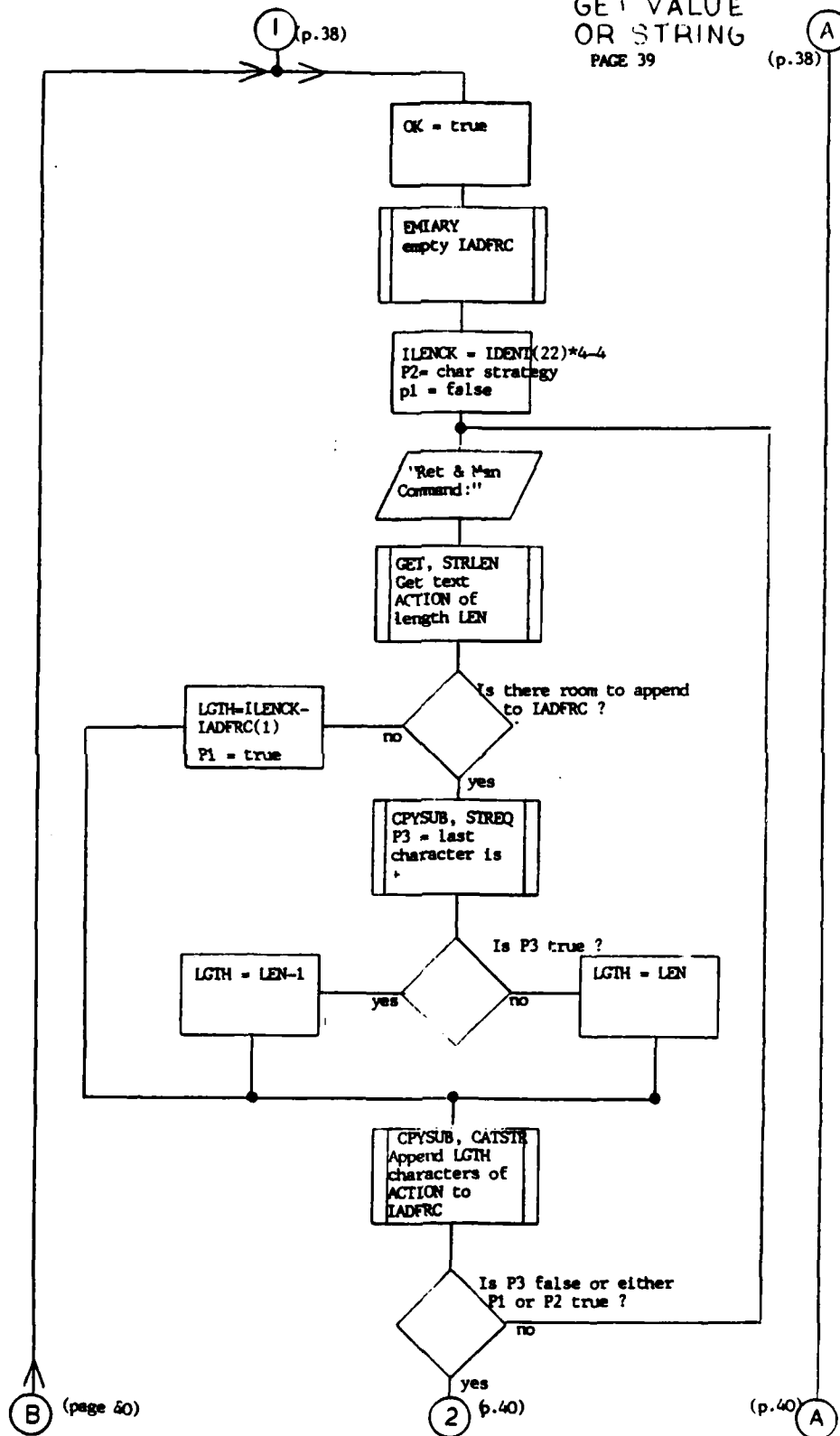
RETEVD  
GET SELECTION  
CRITERIA  
PAGE 37



RETEVD  
GET VALUE  
OR STRING  
PAGE 38

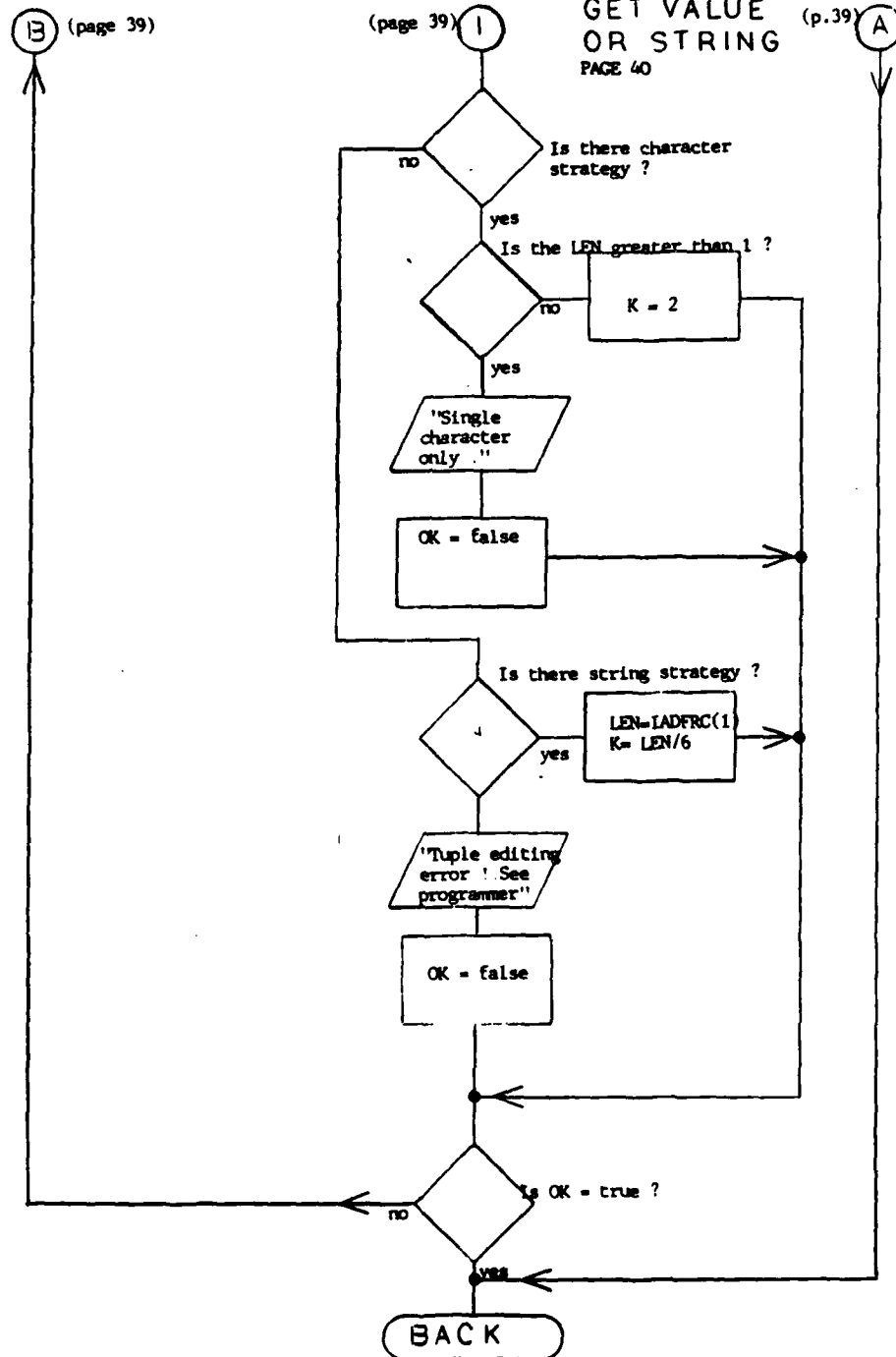


RETEVD  
GET VALUE  
OR STRING  
PAGE 39



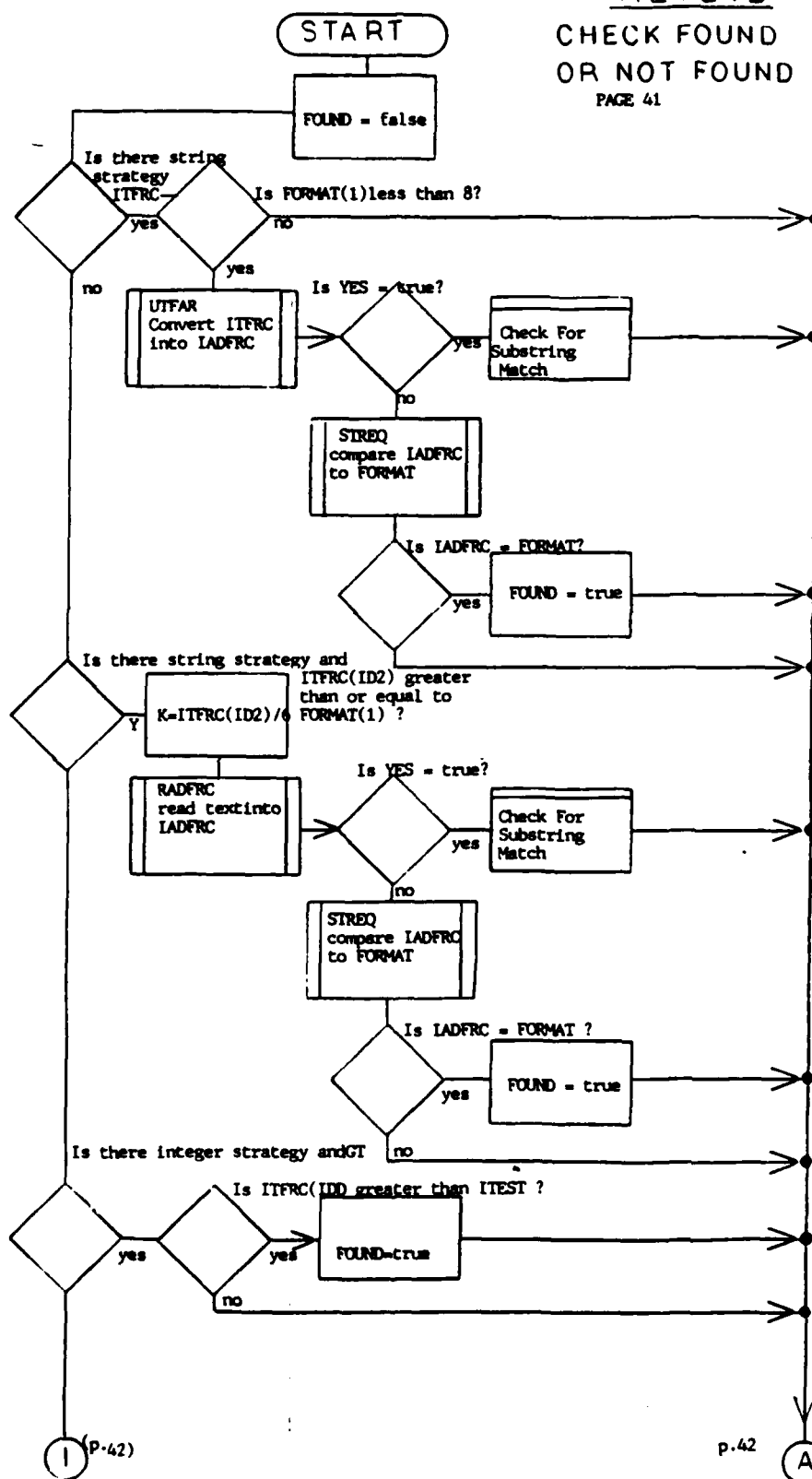
# RETEVD

GET VALUE  
OR STRING  
PAGE 40



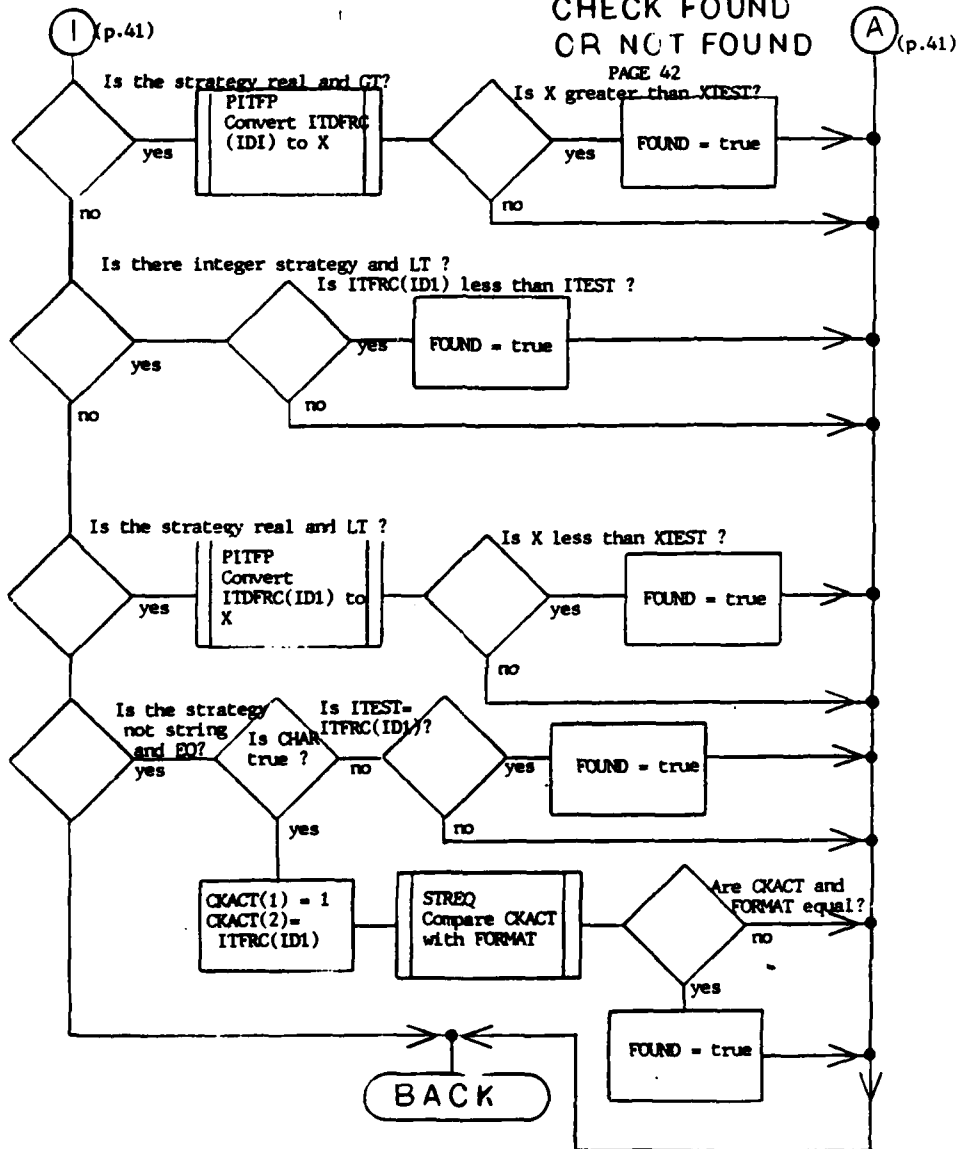


RETEVD  
CHECK FOUND  
OR NOT FOUND  
PAGE 41

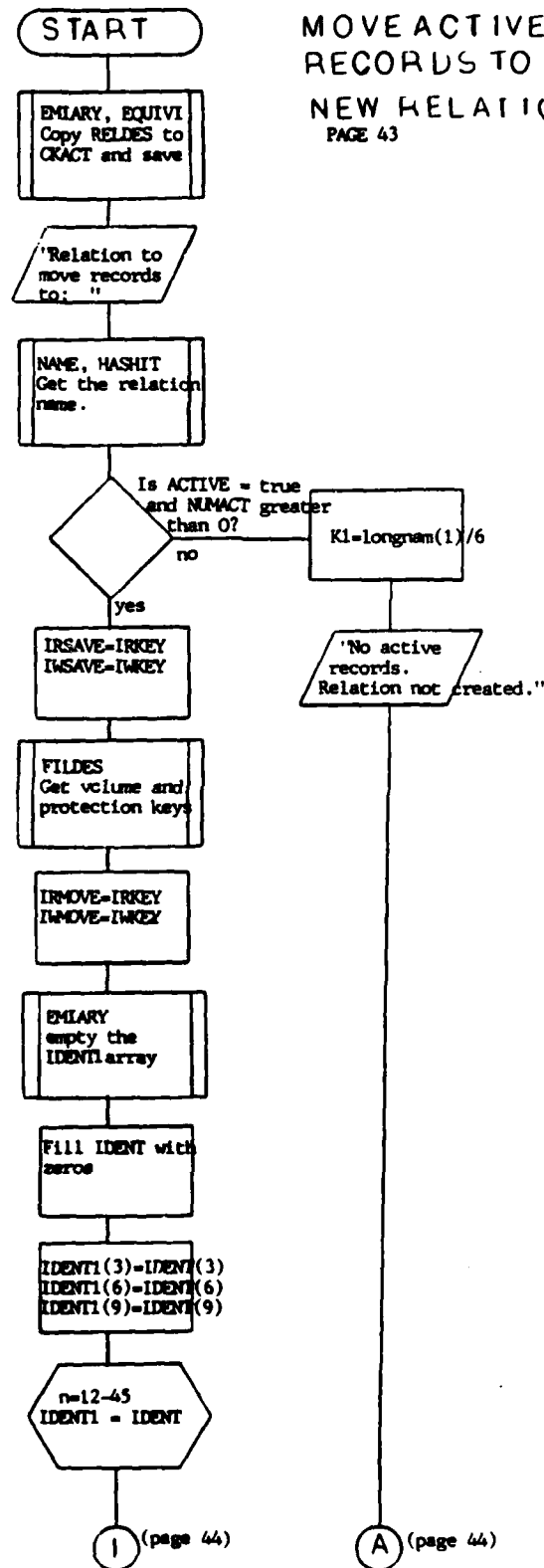


# RETEVD CHECK FOUND OR NOT FOUND

PAGE 42

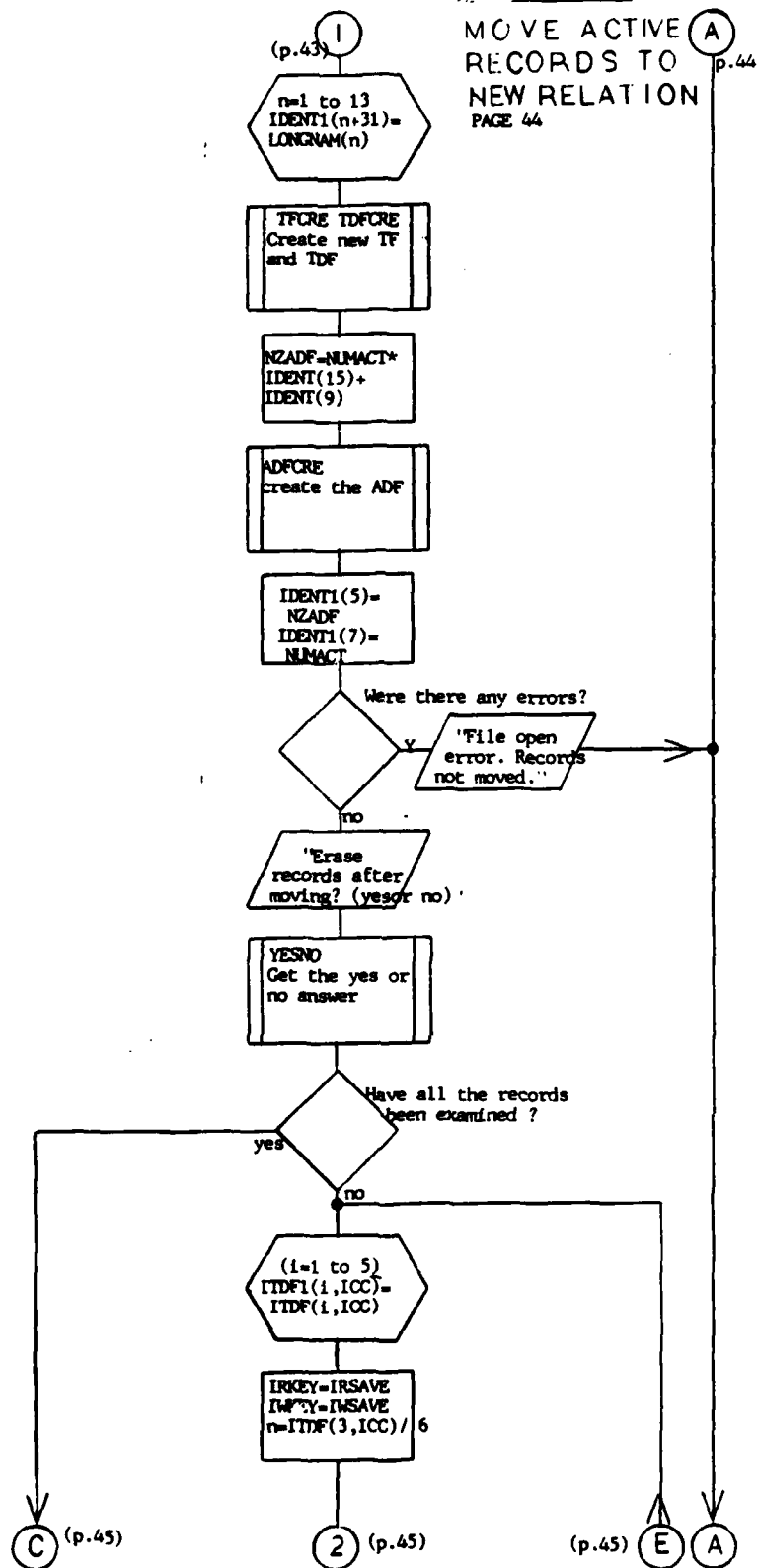


RETEVD  
 MOVE ACTIVE  
 RECORDS TO  
 NEW RELATION  
 PAGE 43

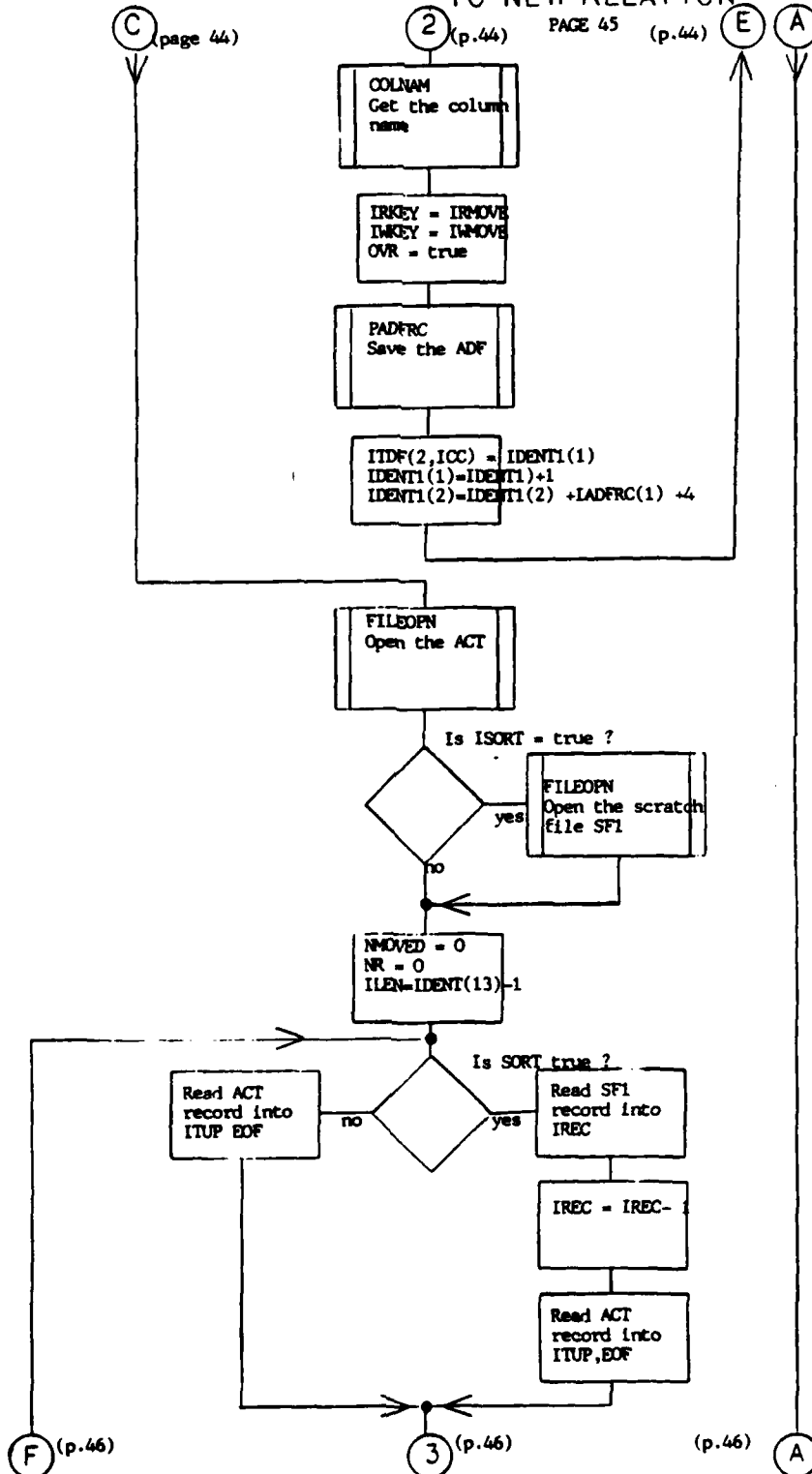


# REL EVD

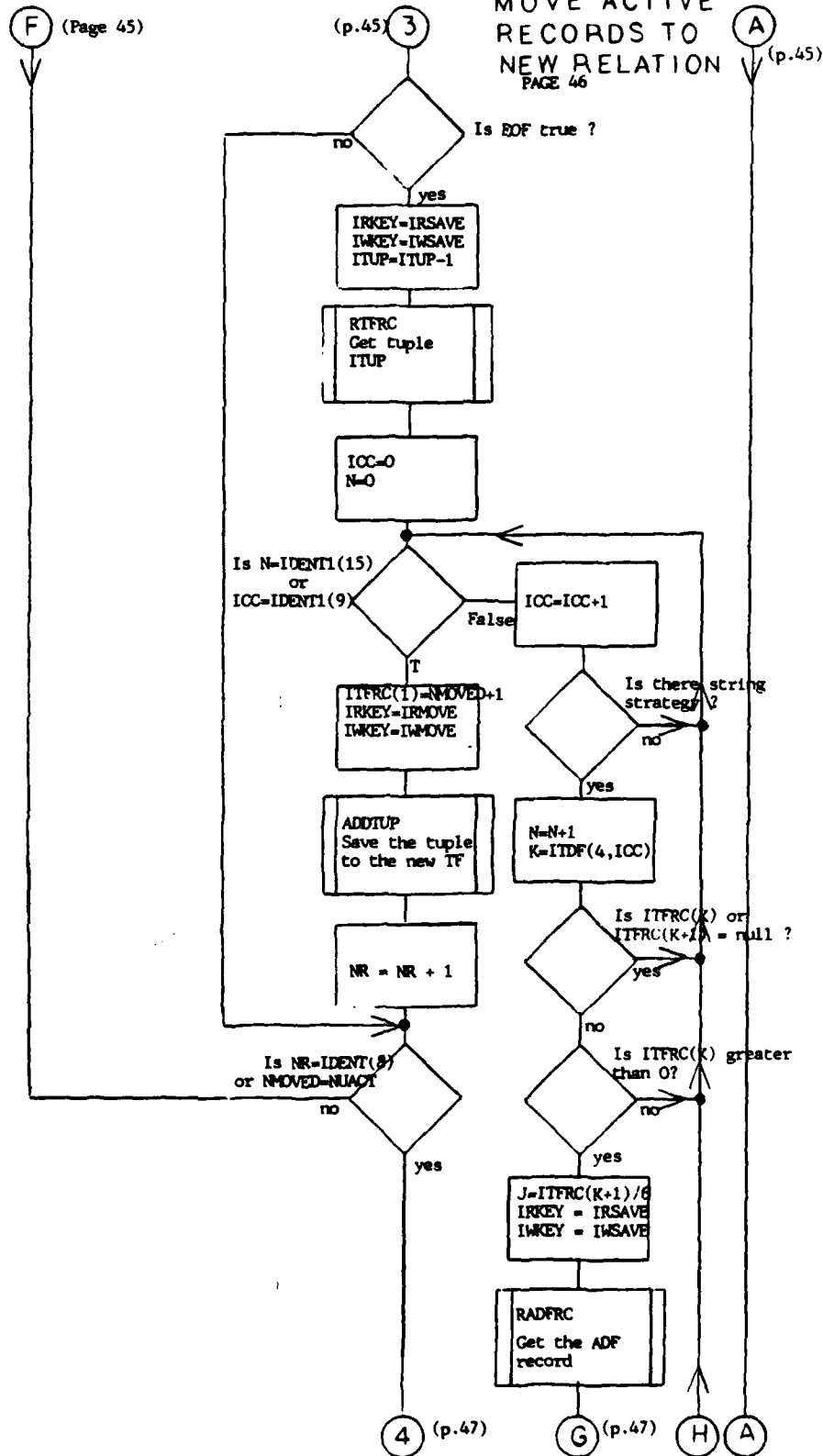
MOVE ACTIVE  
RECORDS TO  
NEW RELATION  
PAGE 44



# RELEVU MOVE ACTIVE RECORDS TO NEW RELATION



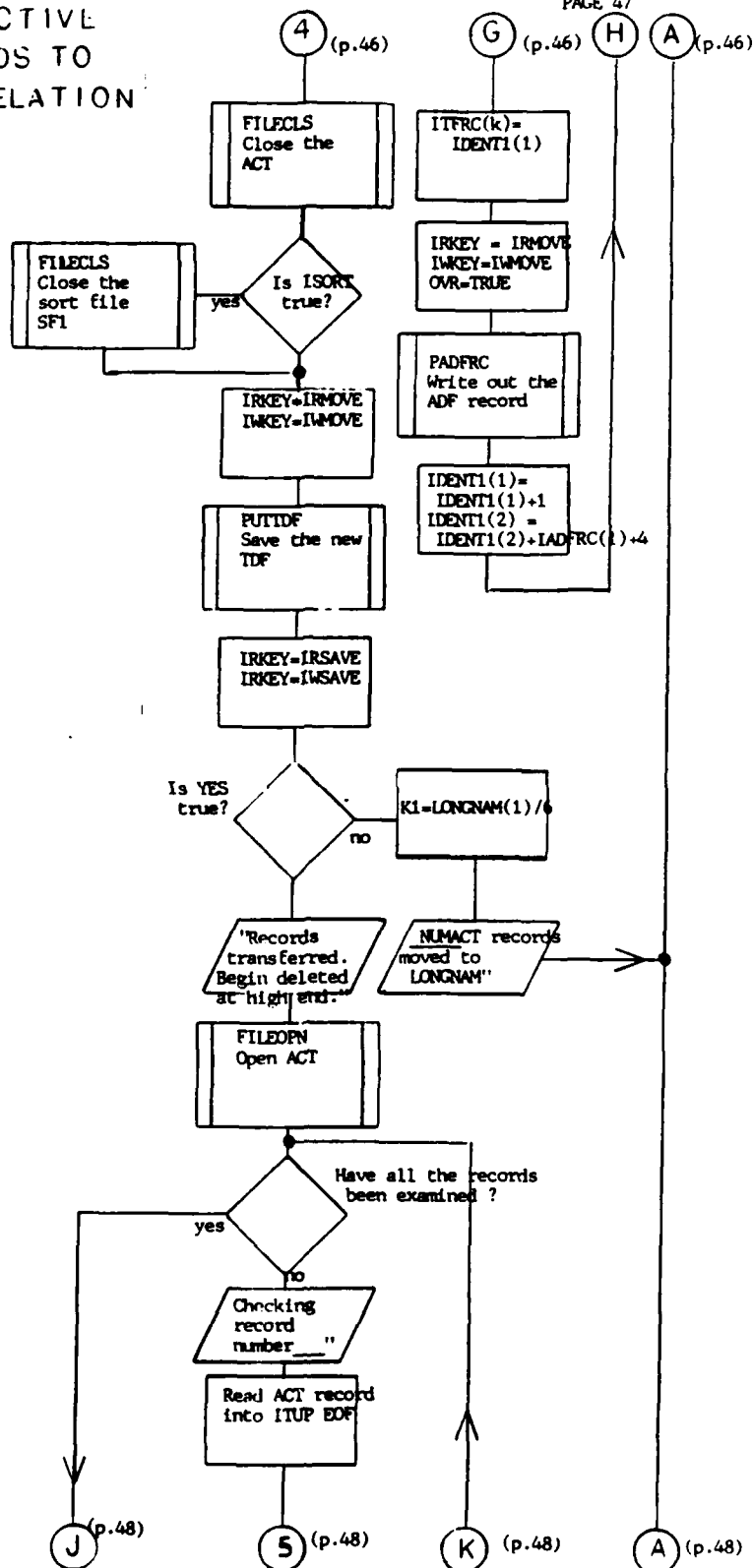
RELVD  
MOVE ACTIVE  
RECORDS TO  
NEW RELATION  
PAGE 46



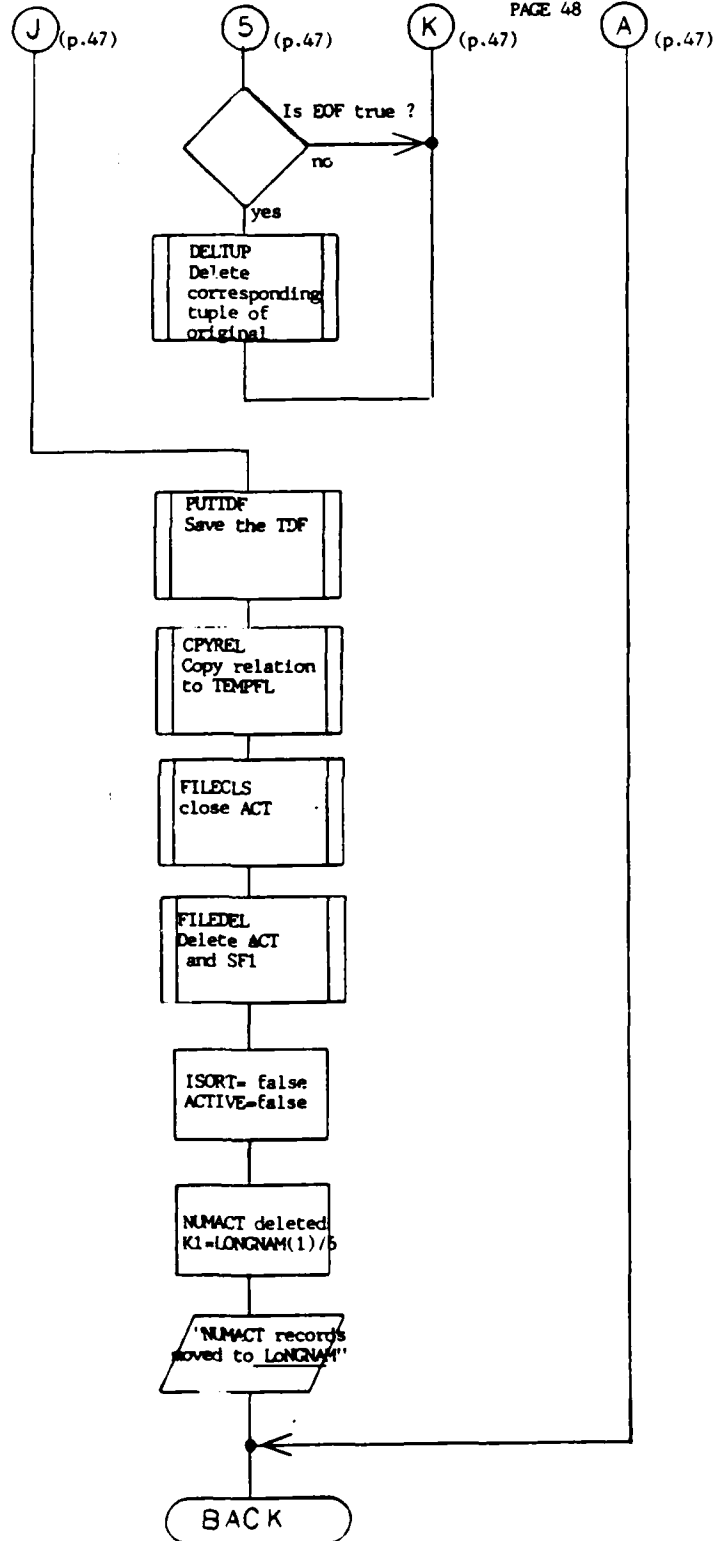
# MOVE ACTIVE RECORDS TO NEW RELATION

KLIEVD

PAGE 47



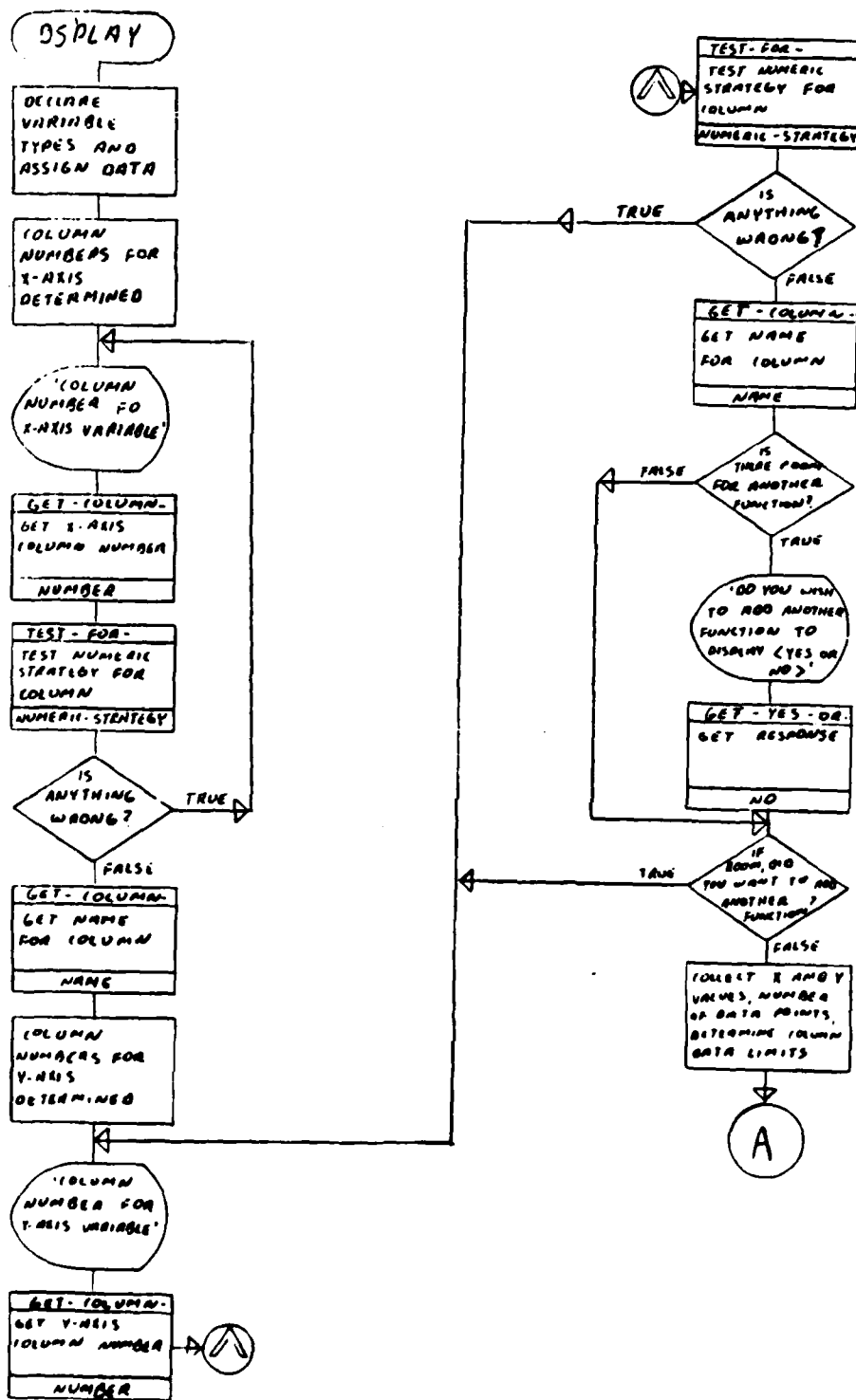
MOVE ACTIVE  
RECORDS TO  
NEW RELATION

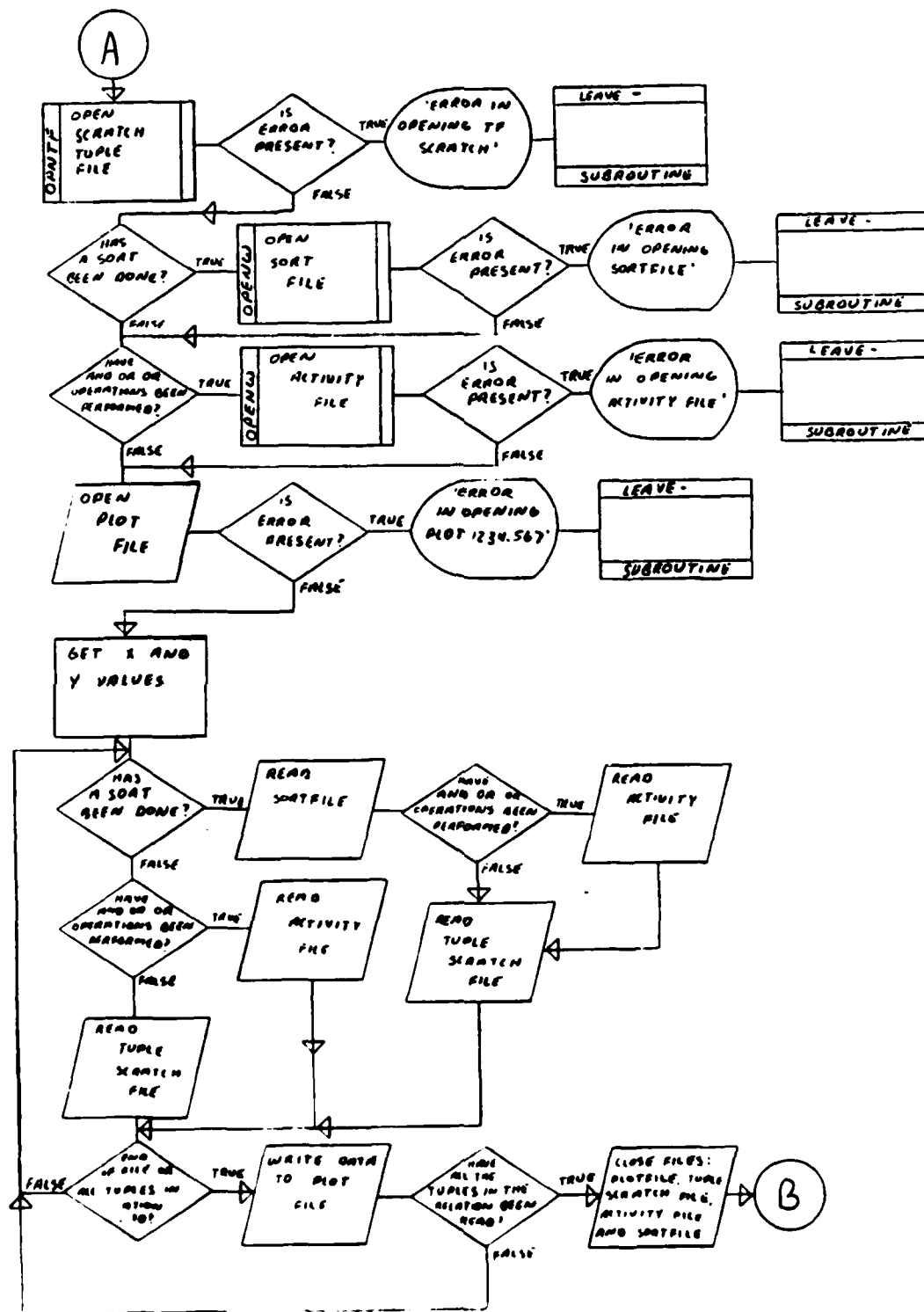


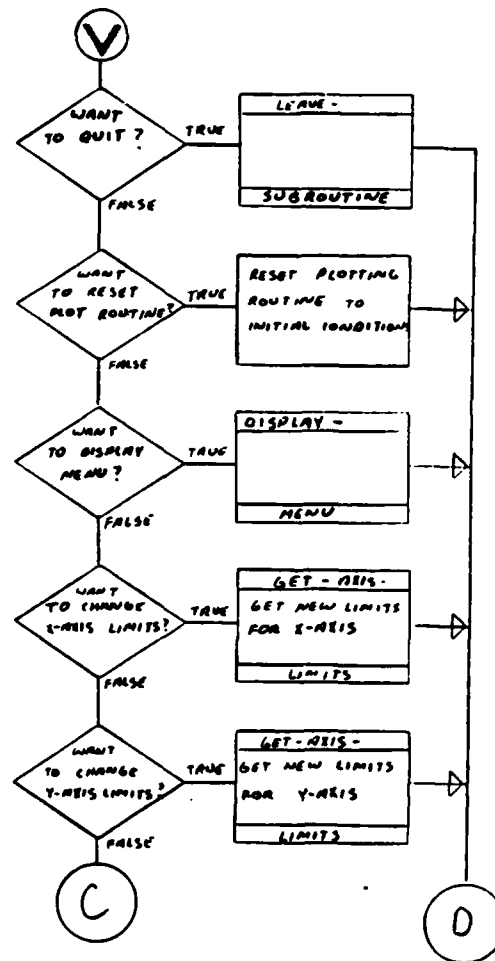
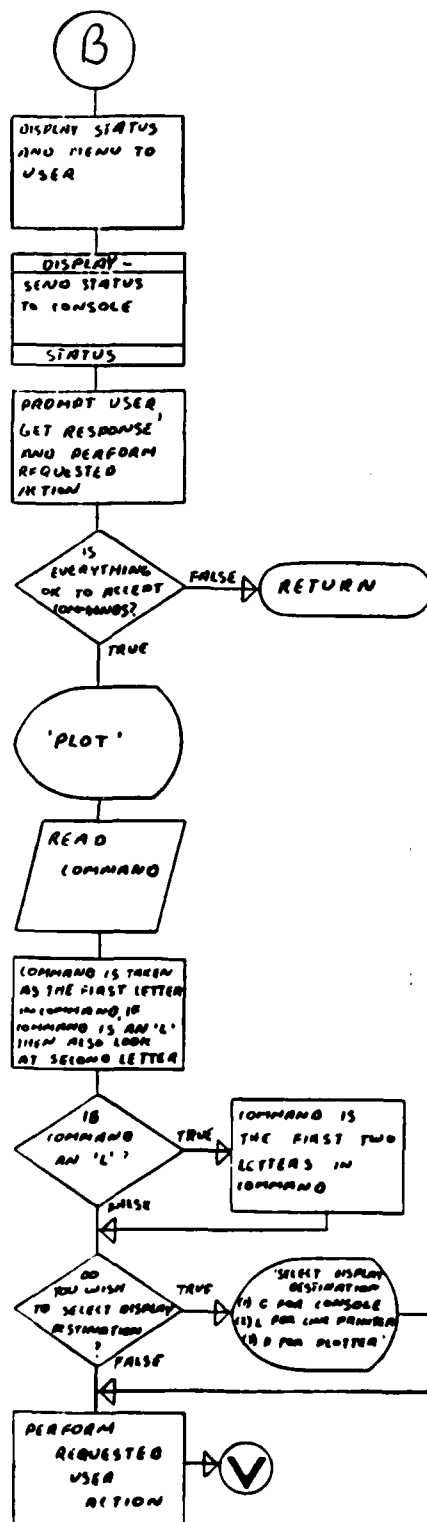


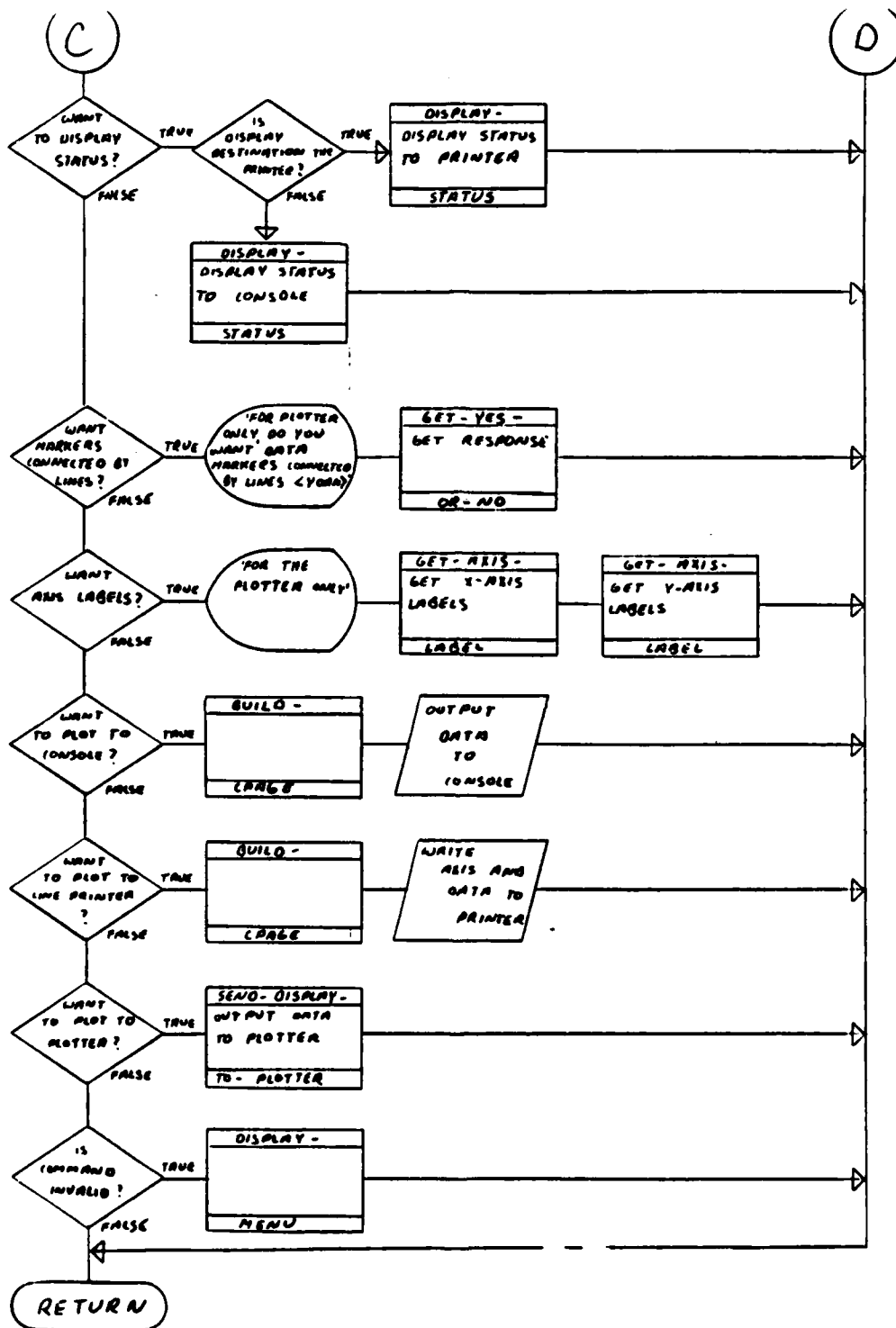
APPENDIX C

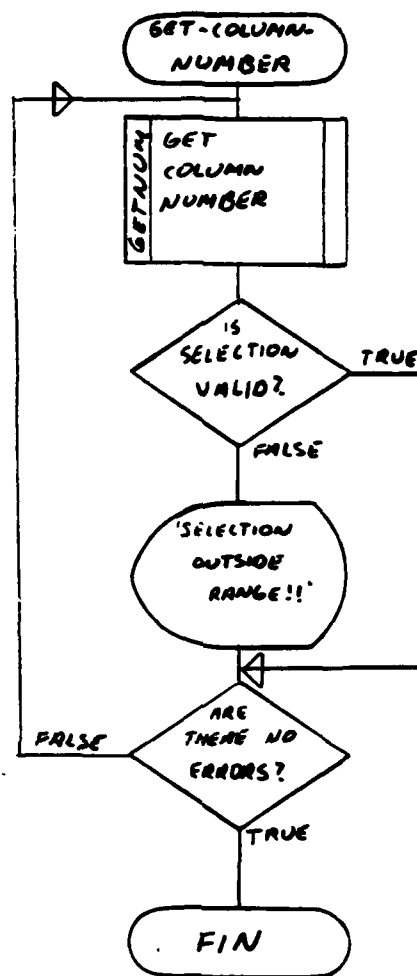
DSPLAY

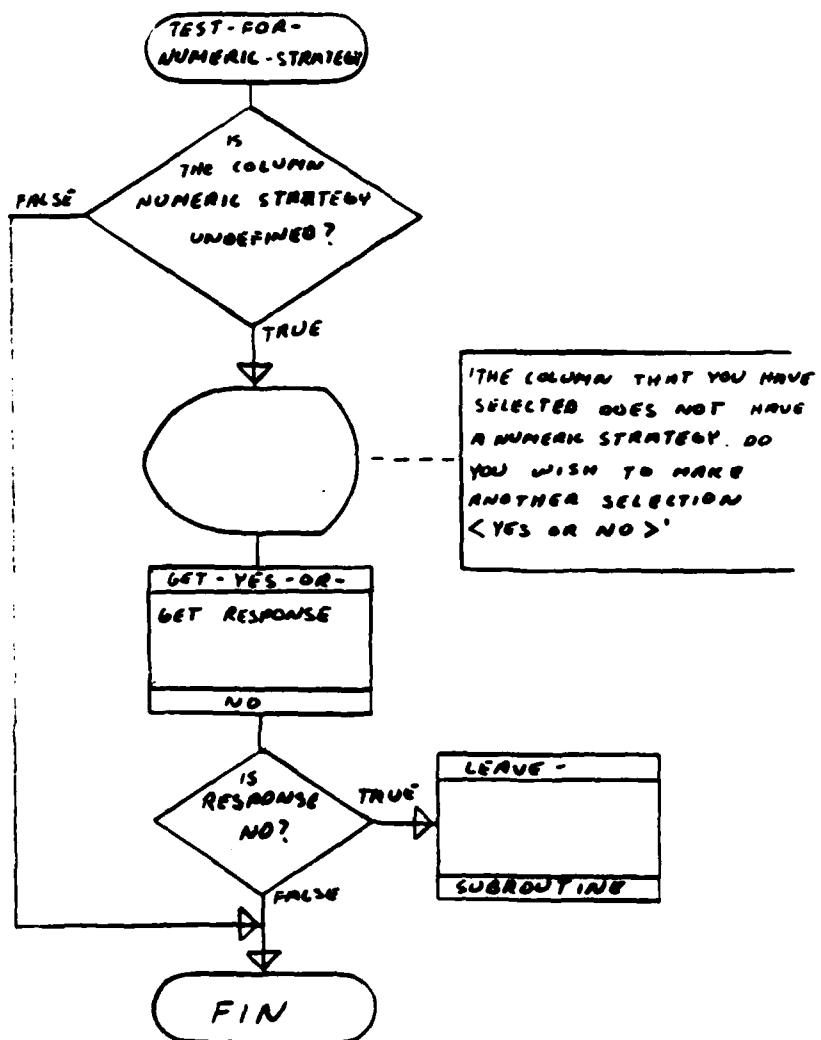


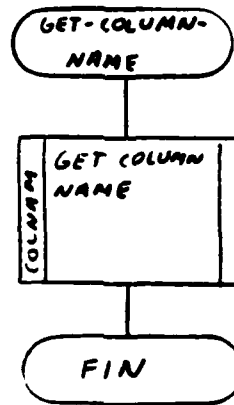




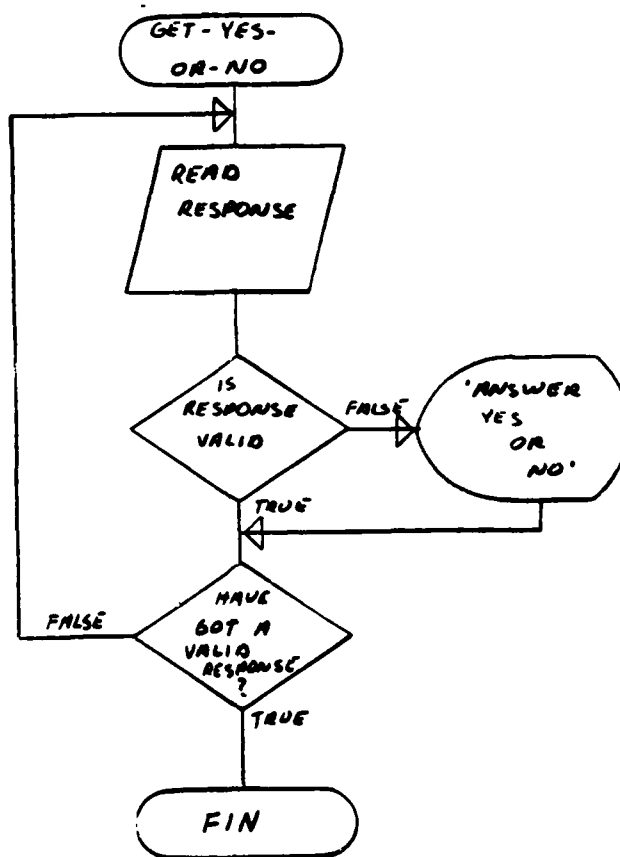


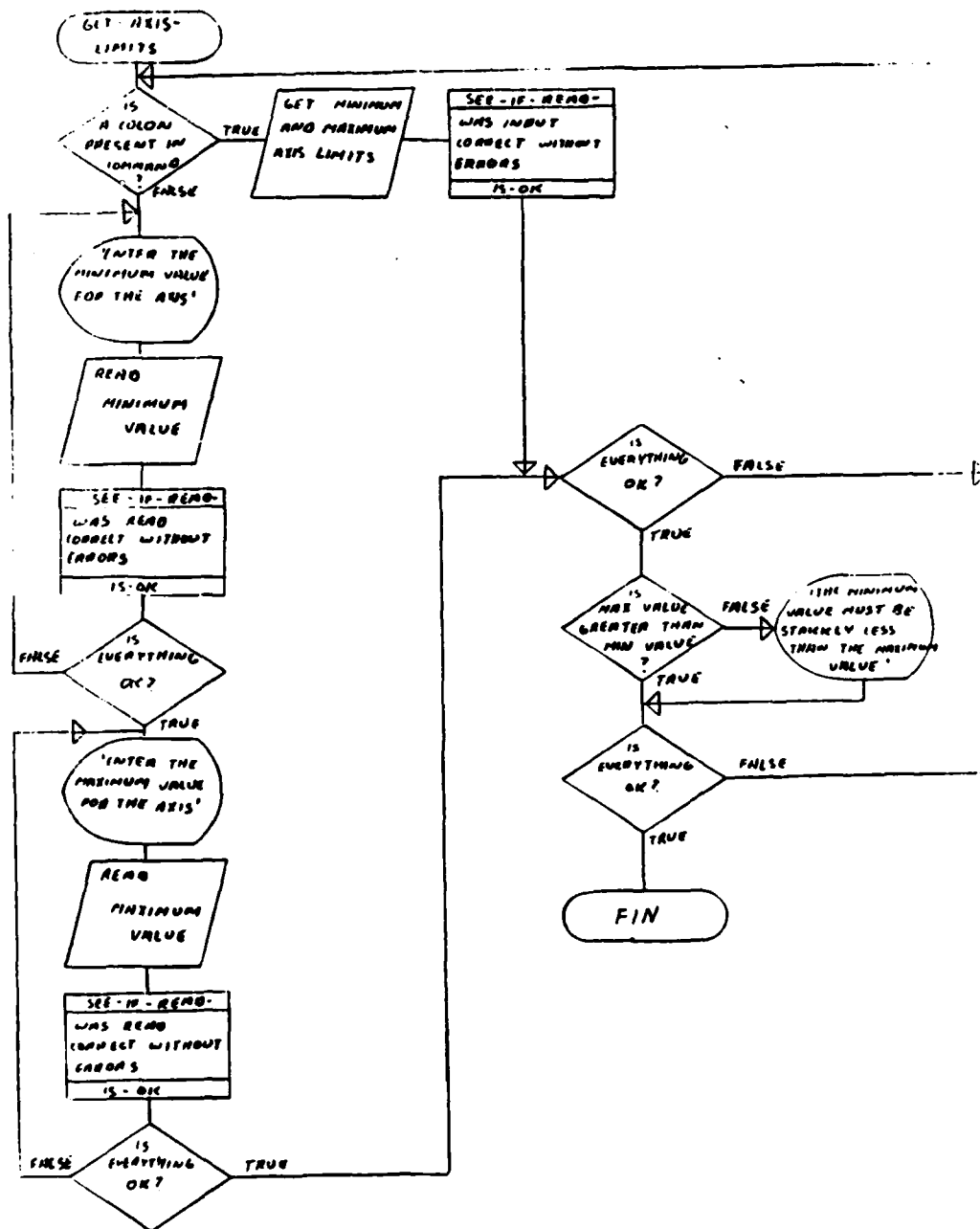


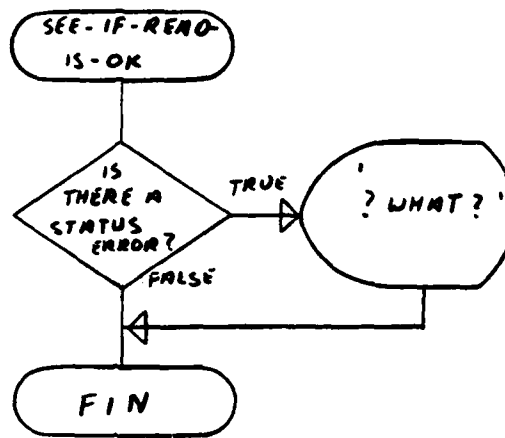


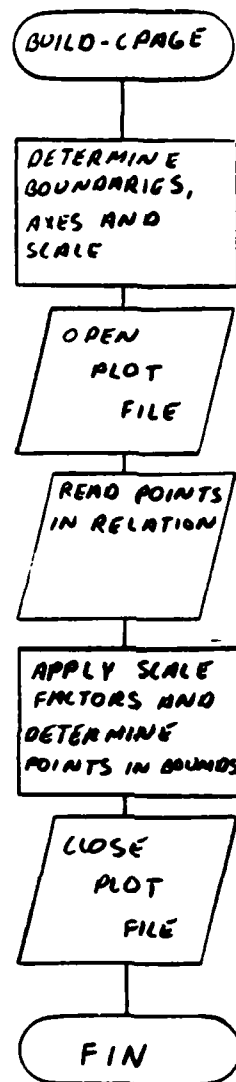


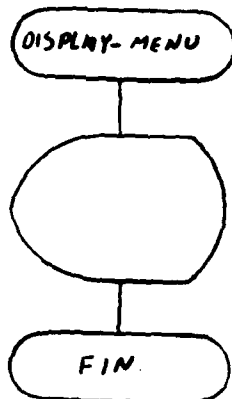






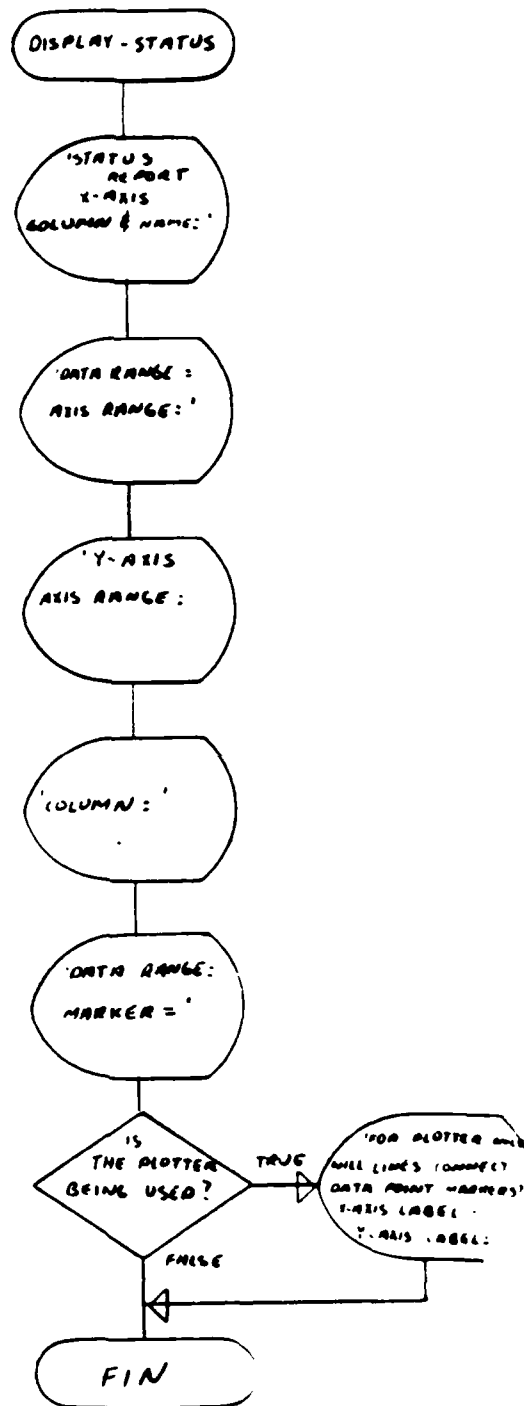






THE LEGAL COMMANDS ARE:

- Q-(QUIT) TERMINATE PLOTTING PROGRAM
- R (RESET) RESET PLOTTING PROGRAM
- S (STATUS) PRINT STATUS REPORT
- X (X-AXIS) SET X-AXIS RANGE
- Y (Y-AXIS) SET Y-AXIS RANGE
- P (PLOT) DISPLAY THE PLOT
- ? LIST THE LEGAL COMMANDS
- LL (LINE) (DRAW) THE DATA POINTS WITH LINES
- LA (LABEL) (CREATE X AND Y AXIS LABELS



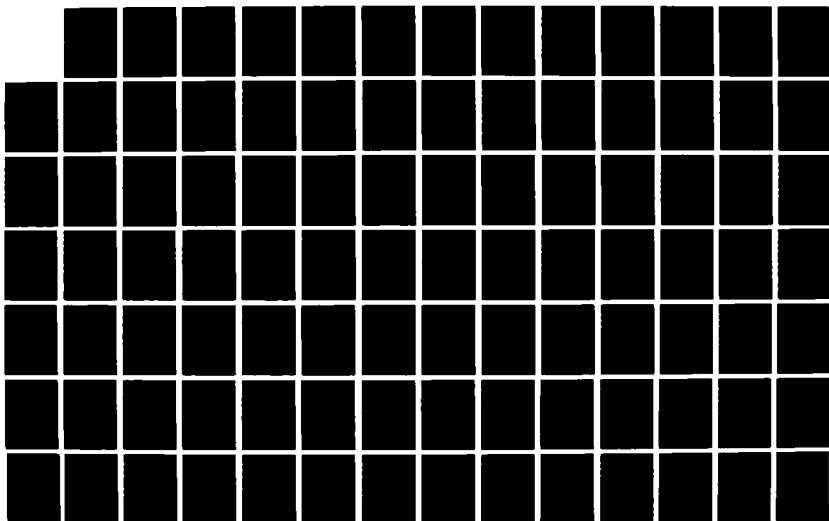
NO-A180 002

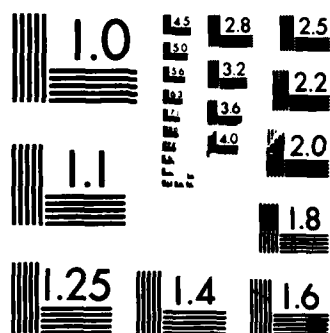
FEASIL IMPLEMENTATION UNDER VAX VMS WITH DESIGN  
INFORMATION(U) ALABAMA UNIV IN HUNTSVILLE DEPT OF  
ELECTRICAL AND COMPUTER EN. J D HARR ET AL. NOV 86  
UAM-5-31325 AMSMI-CR-RD-SS-86-5 F/G 12/5

2/4

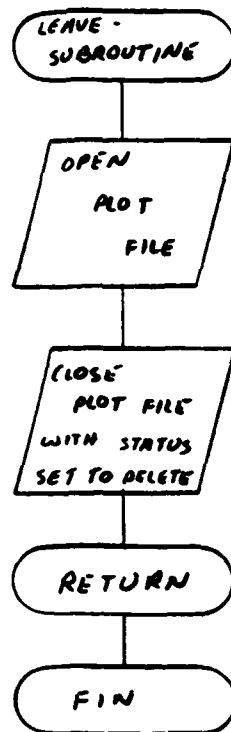
UNCLASSIFIED

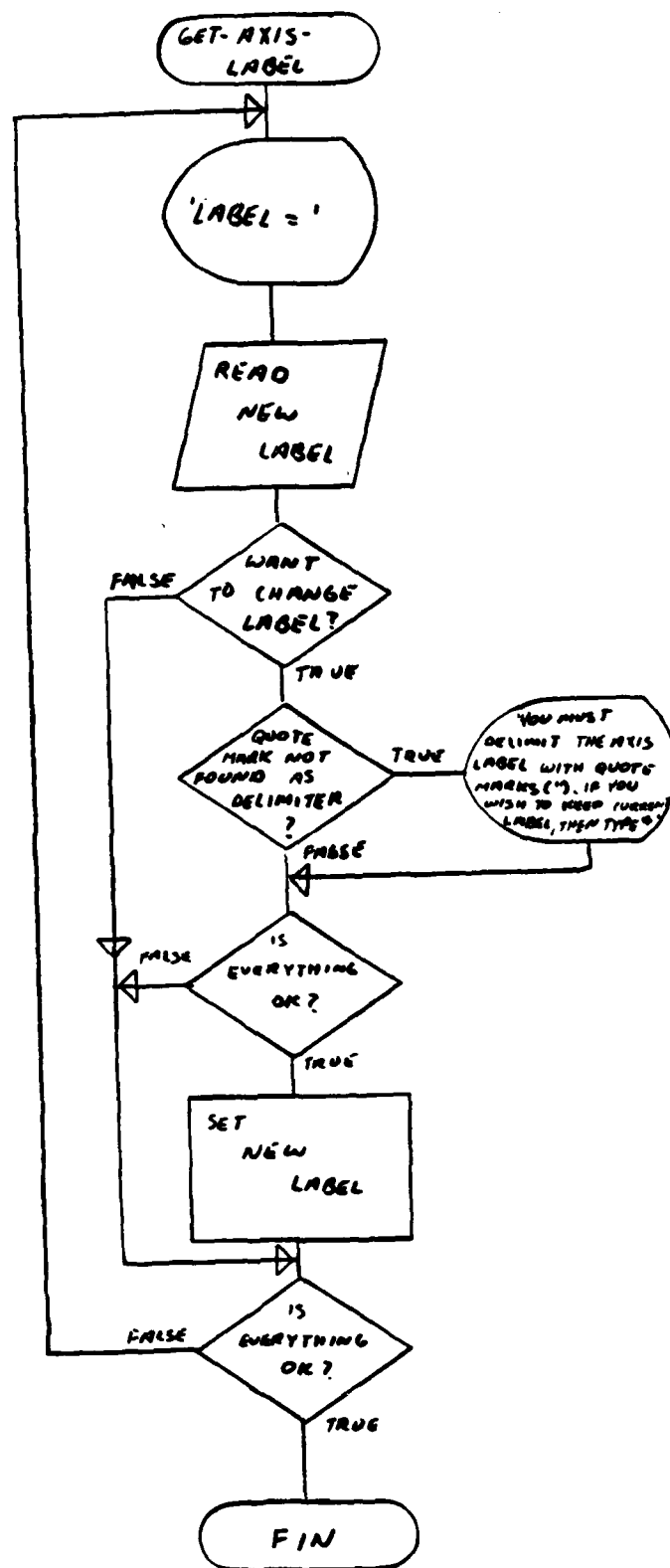
NN

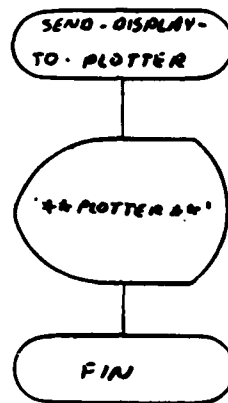












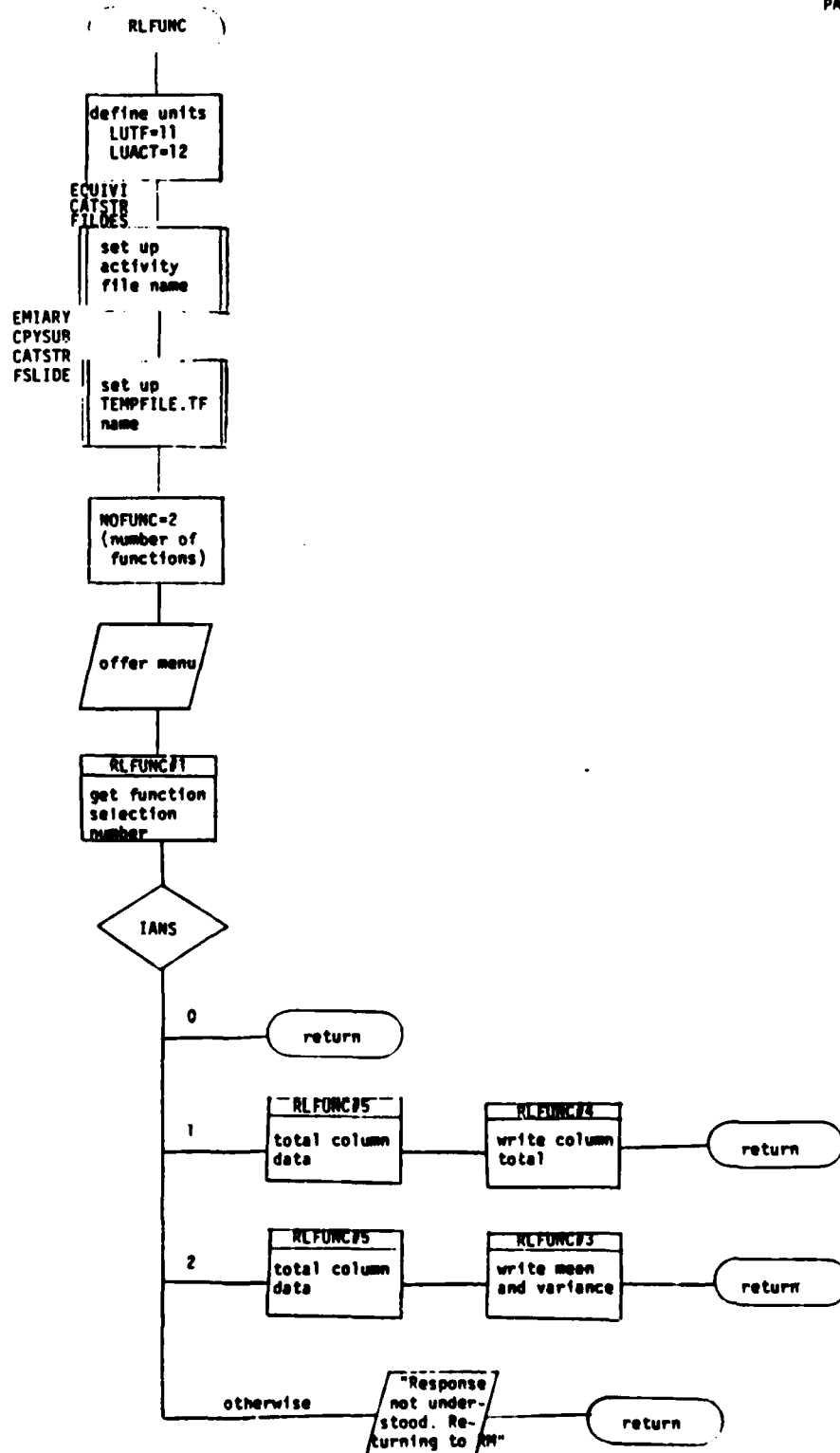
APPENDIX D

RLFUNC

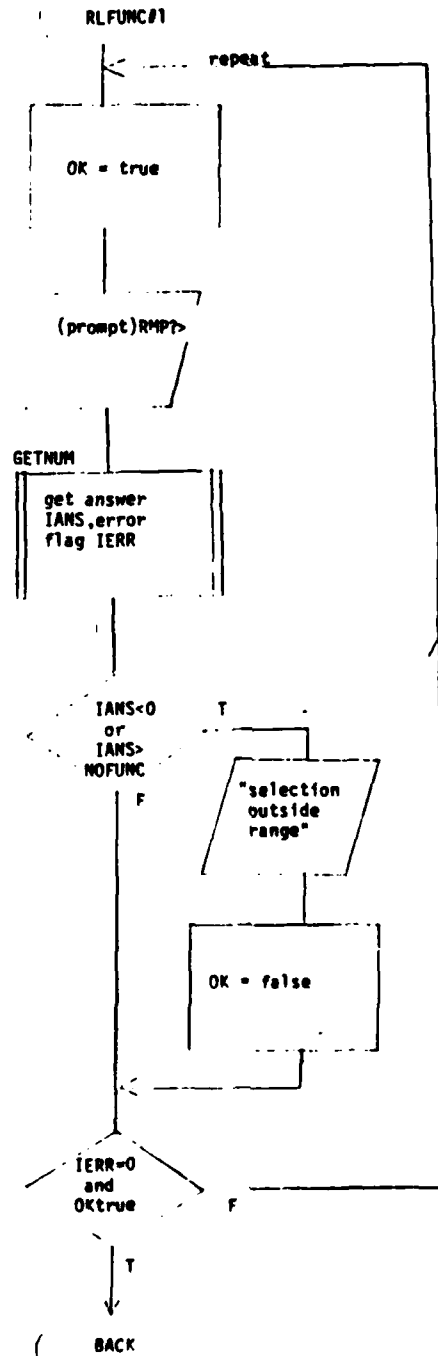
Subroutine : FUNCTION (Rlfunc)

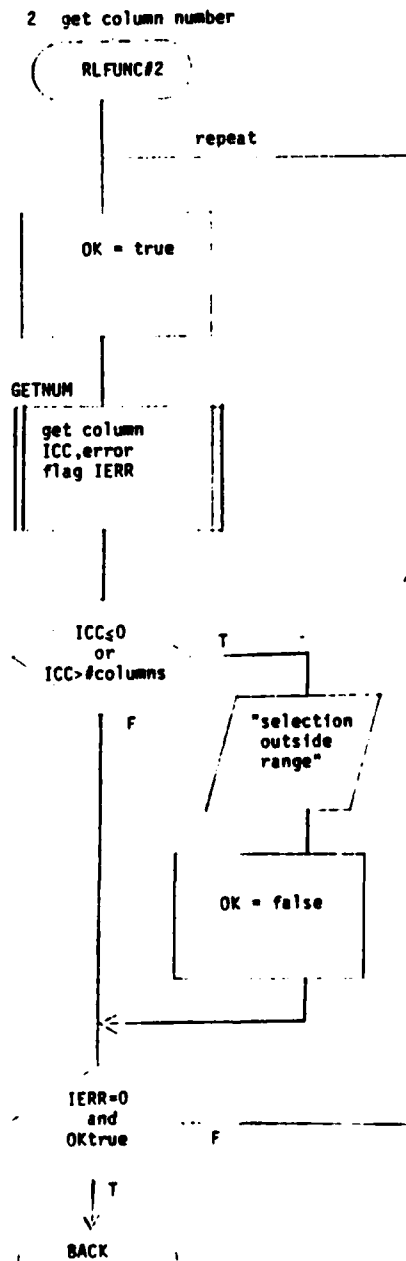
This subroutine provides the user with data analysis functions to allow analysis of sum of column, averaging, mean and standard deviation, etc. This subroutine is constructed that more functions may be easily added as the need arises. The functions available are:

1. Total Column Data      This function algebraically adds the active records of a relation for the column specified. The strategy must be integer or floating point.
2. Mean and Variance      This function computes the mean and variance of the active records in the specified column. Output is the mean and standard deviation.



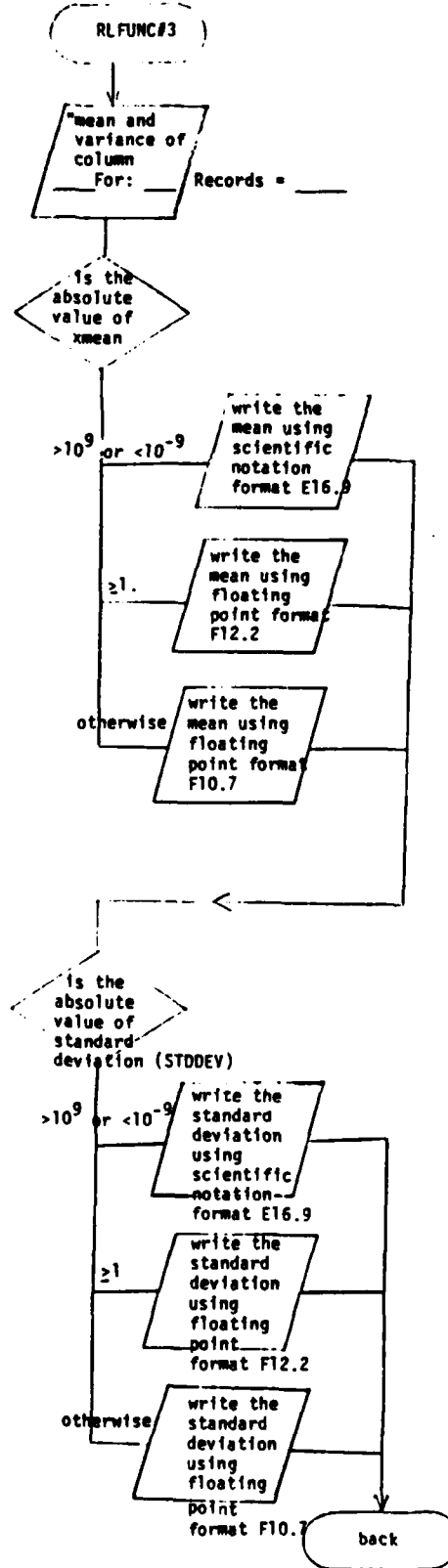
get function selection number



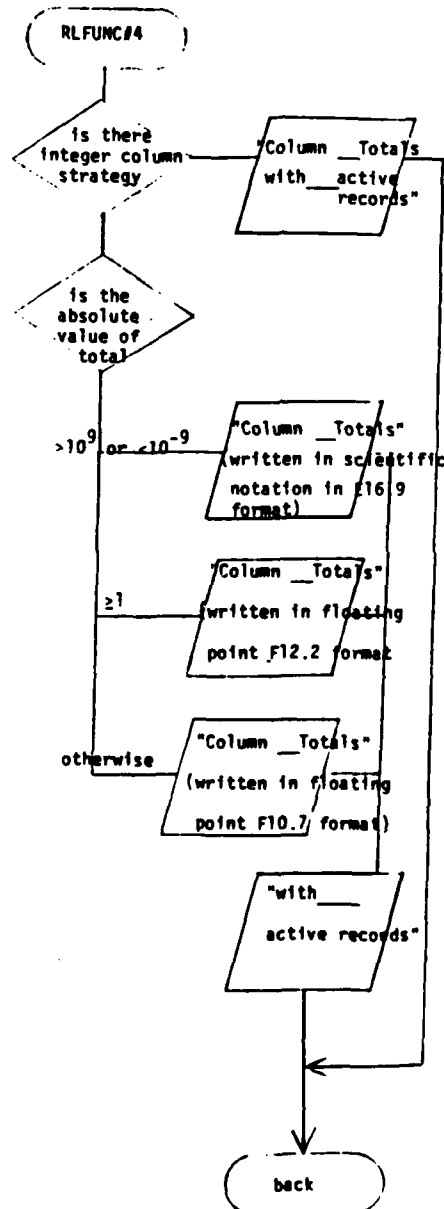


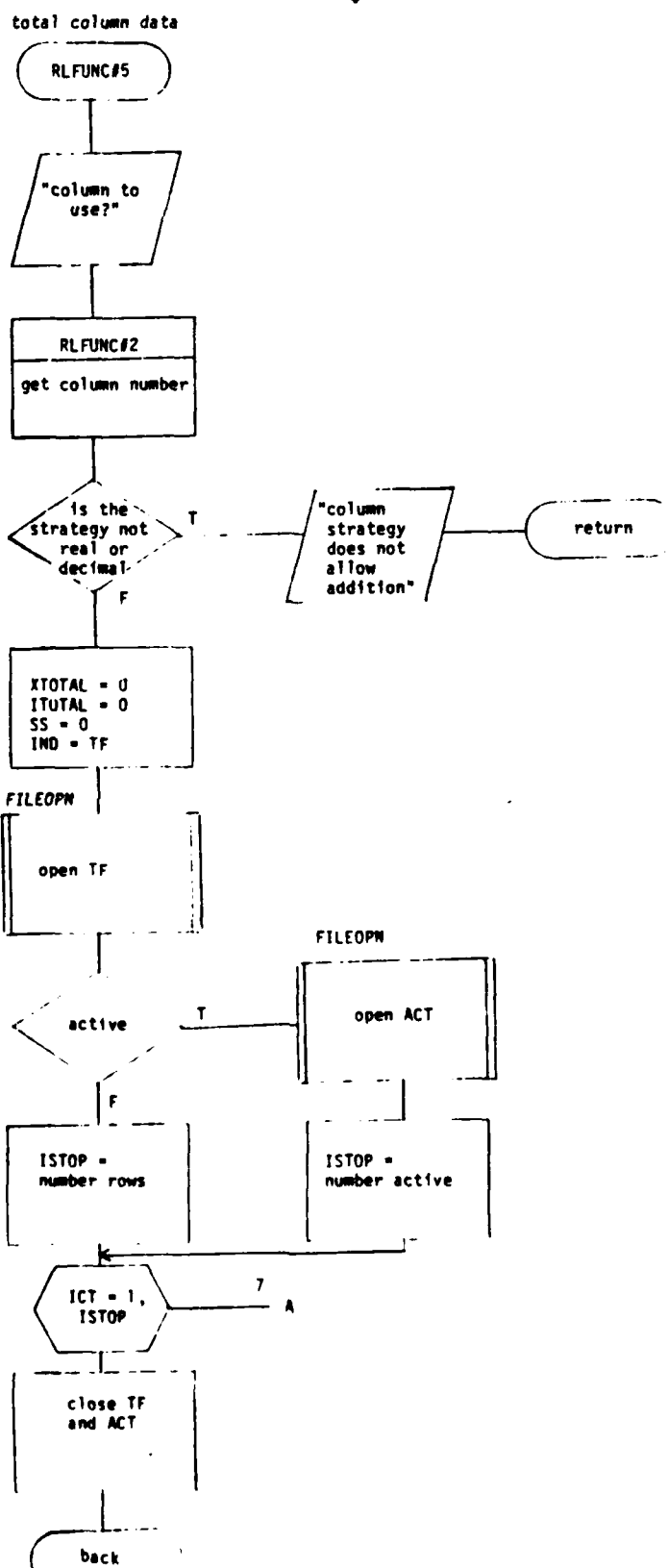


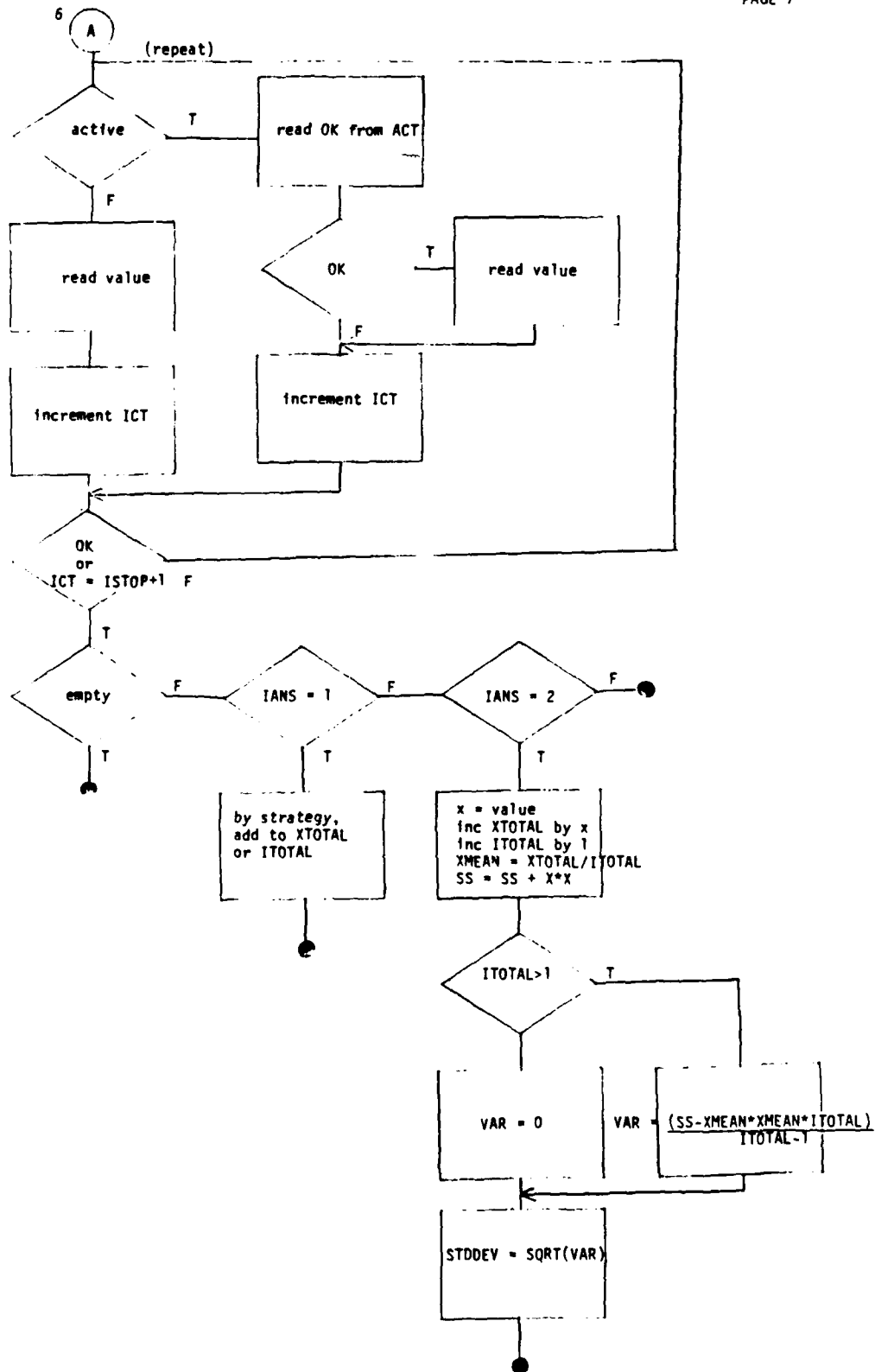
write column mean and variance



write column total

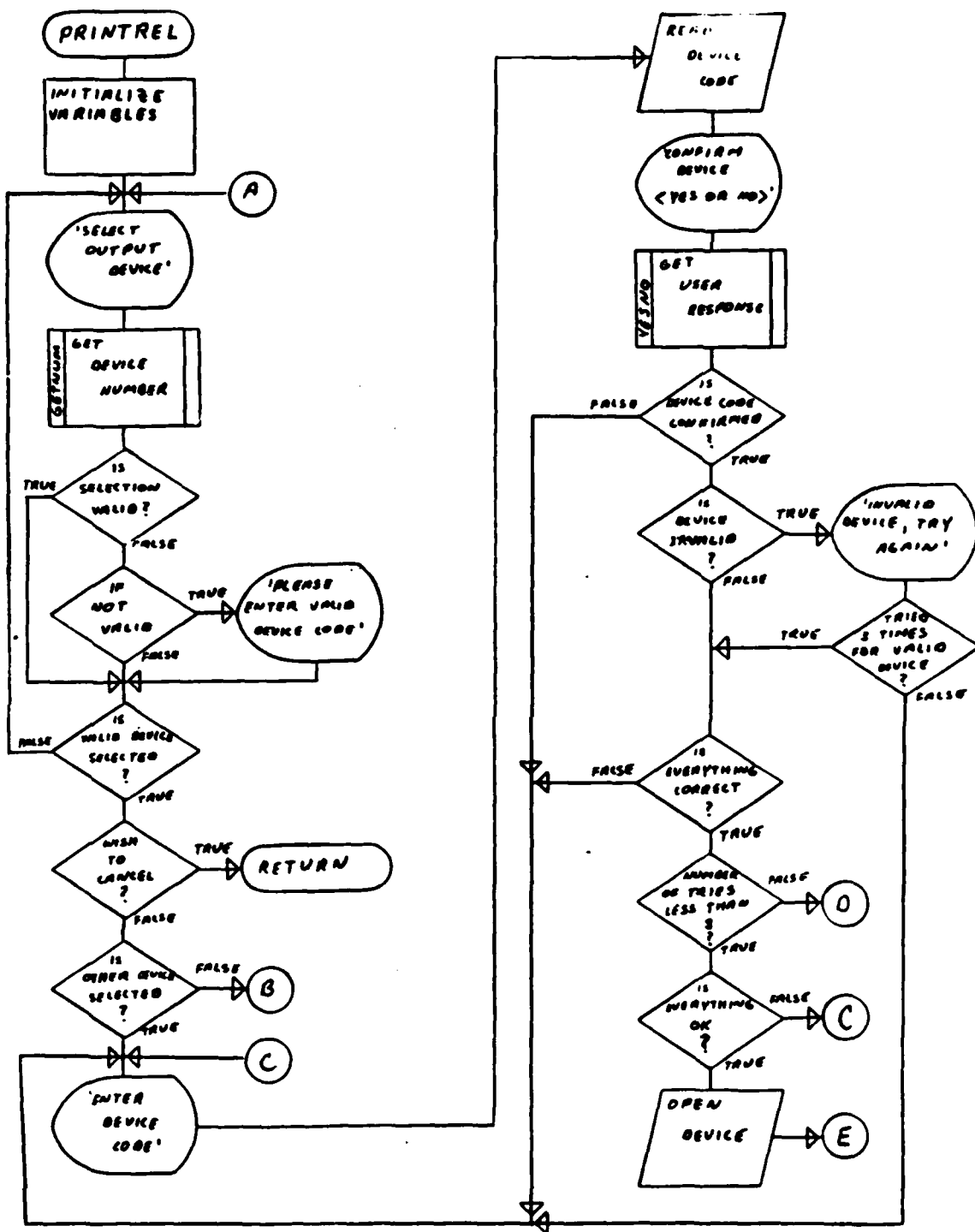


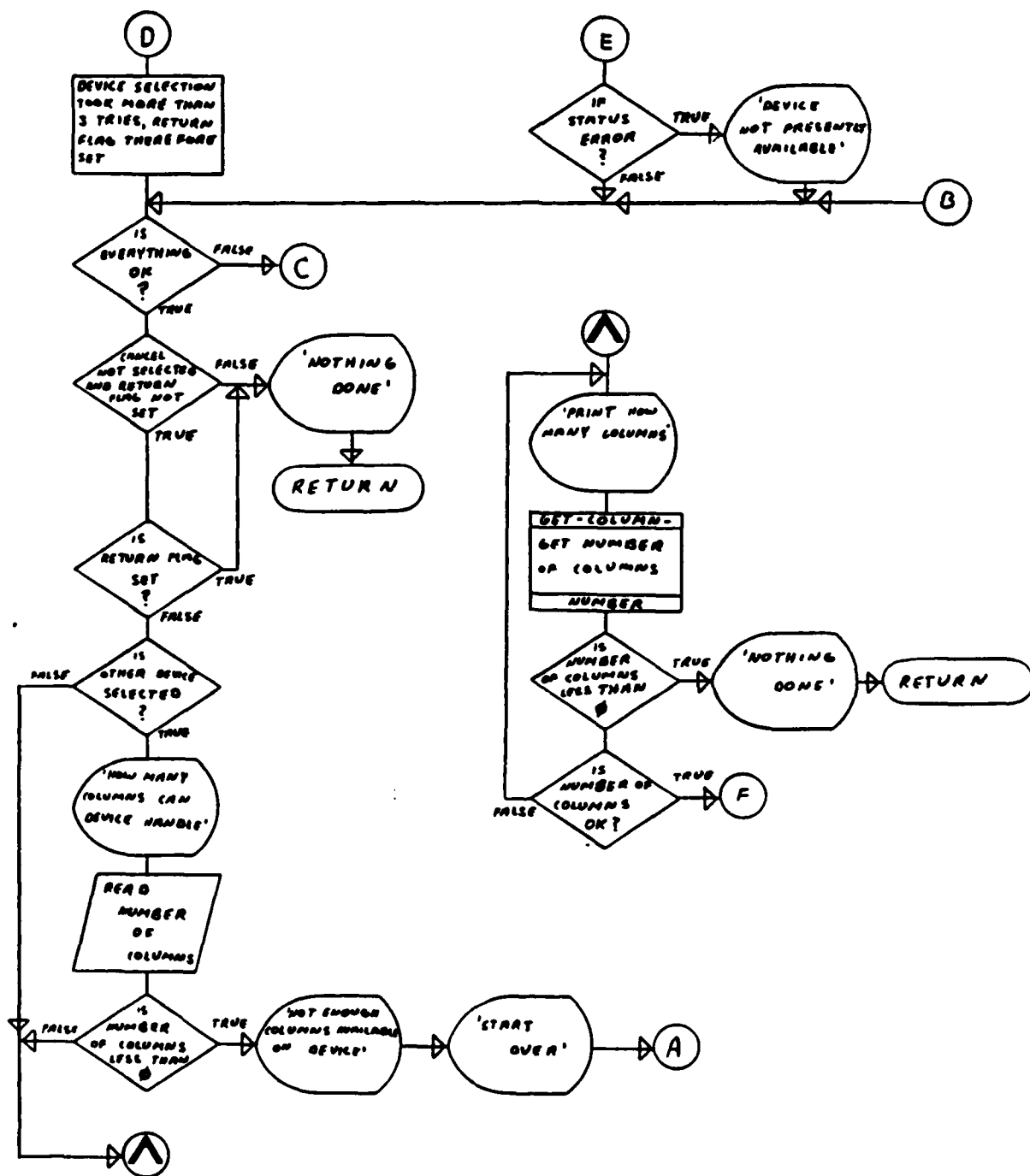


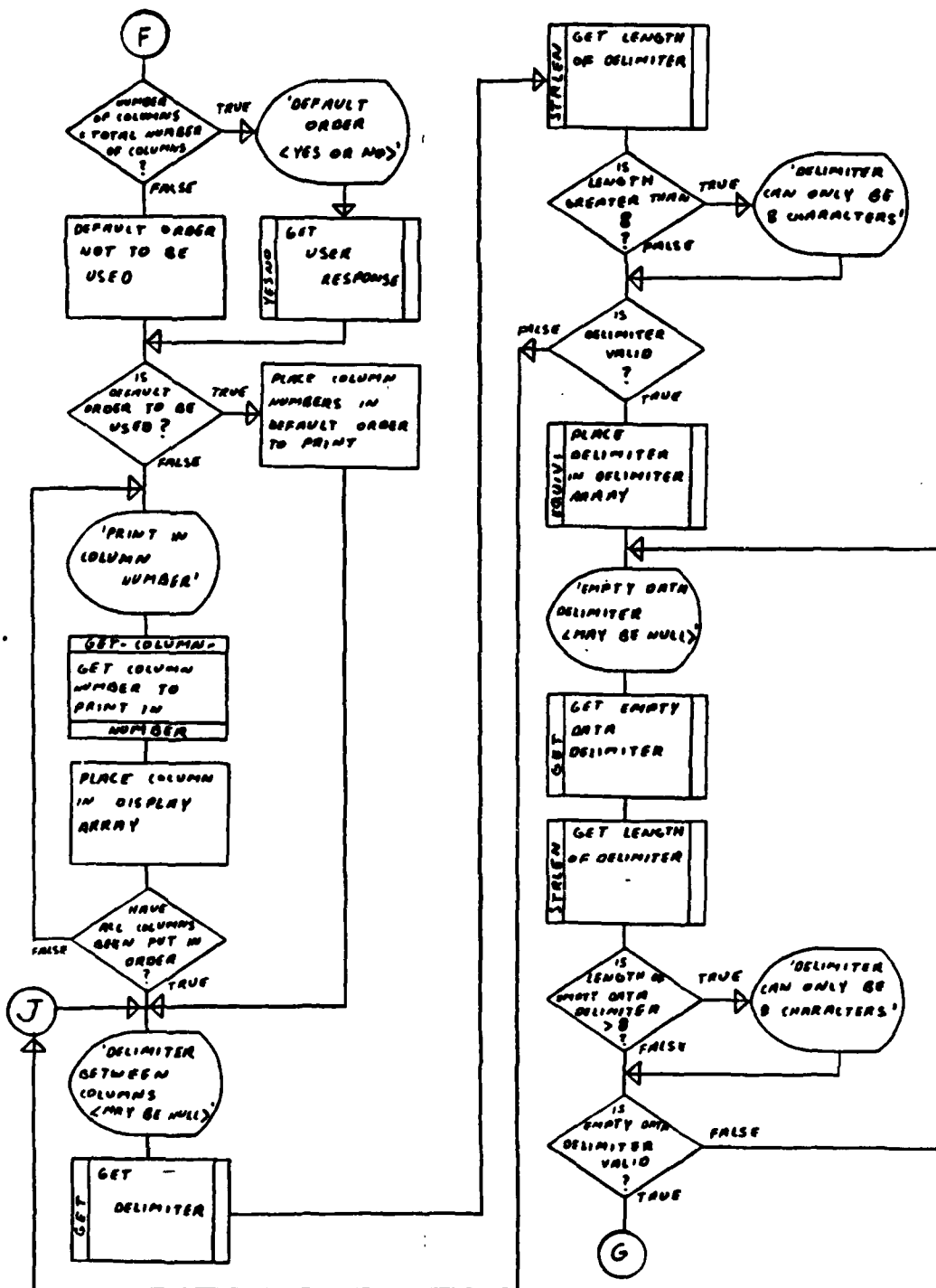


APPENDIX E

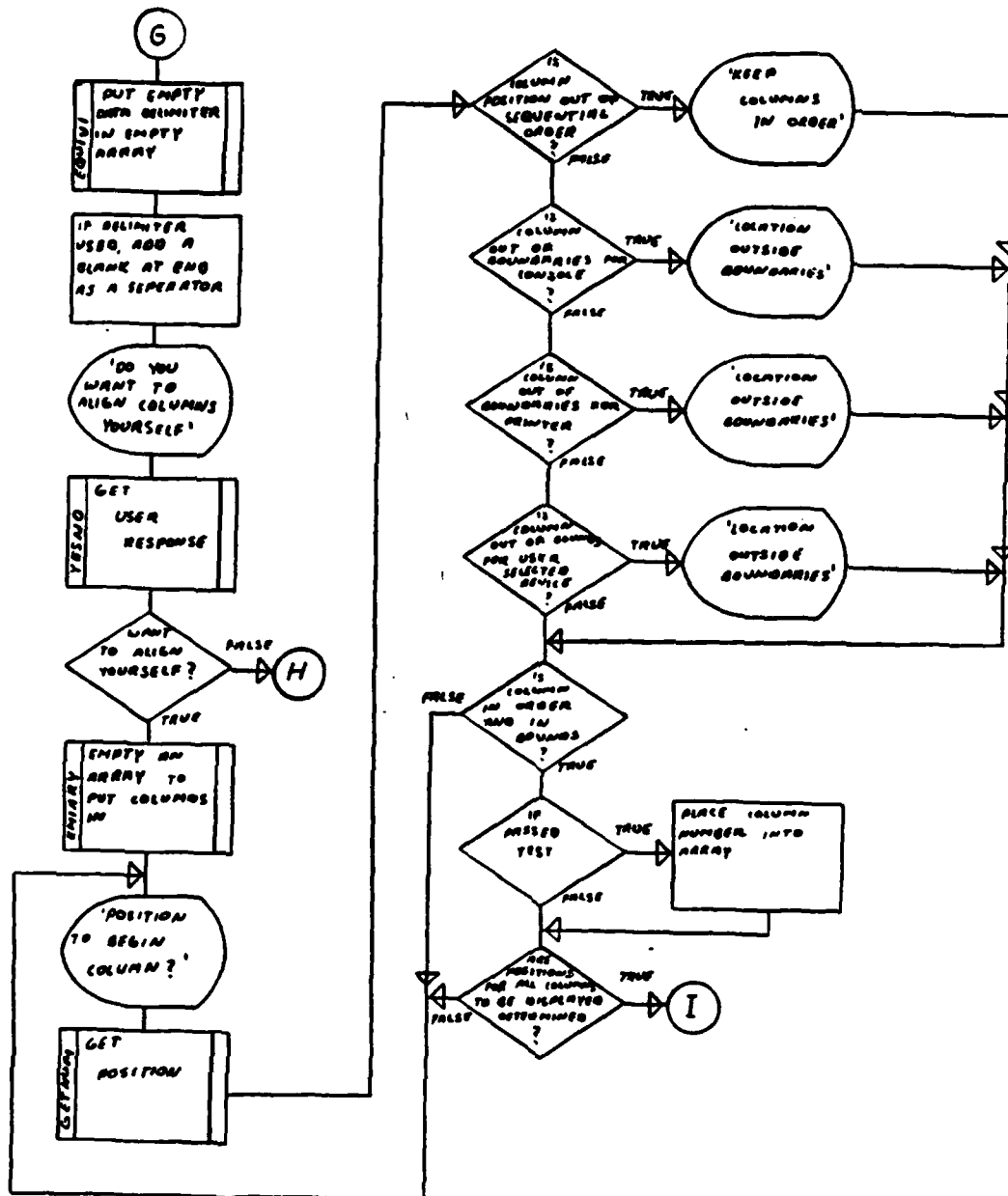
PRINTREL

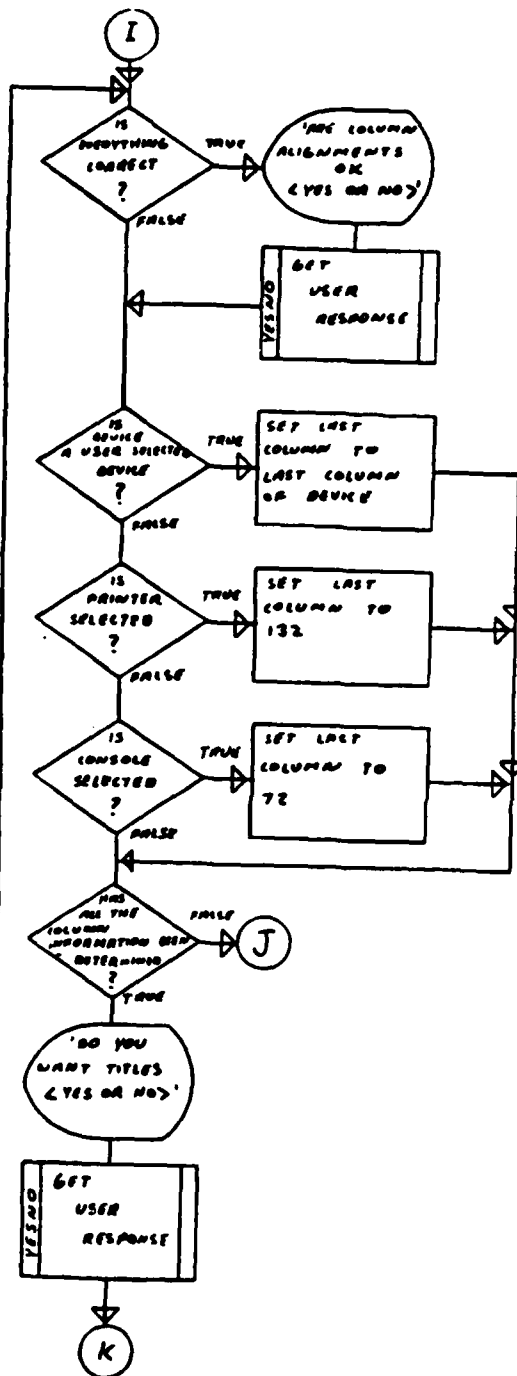
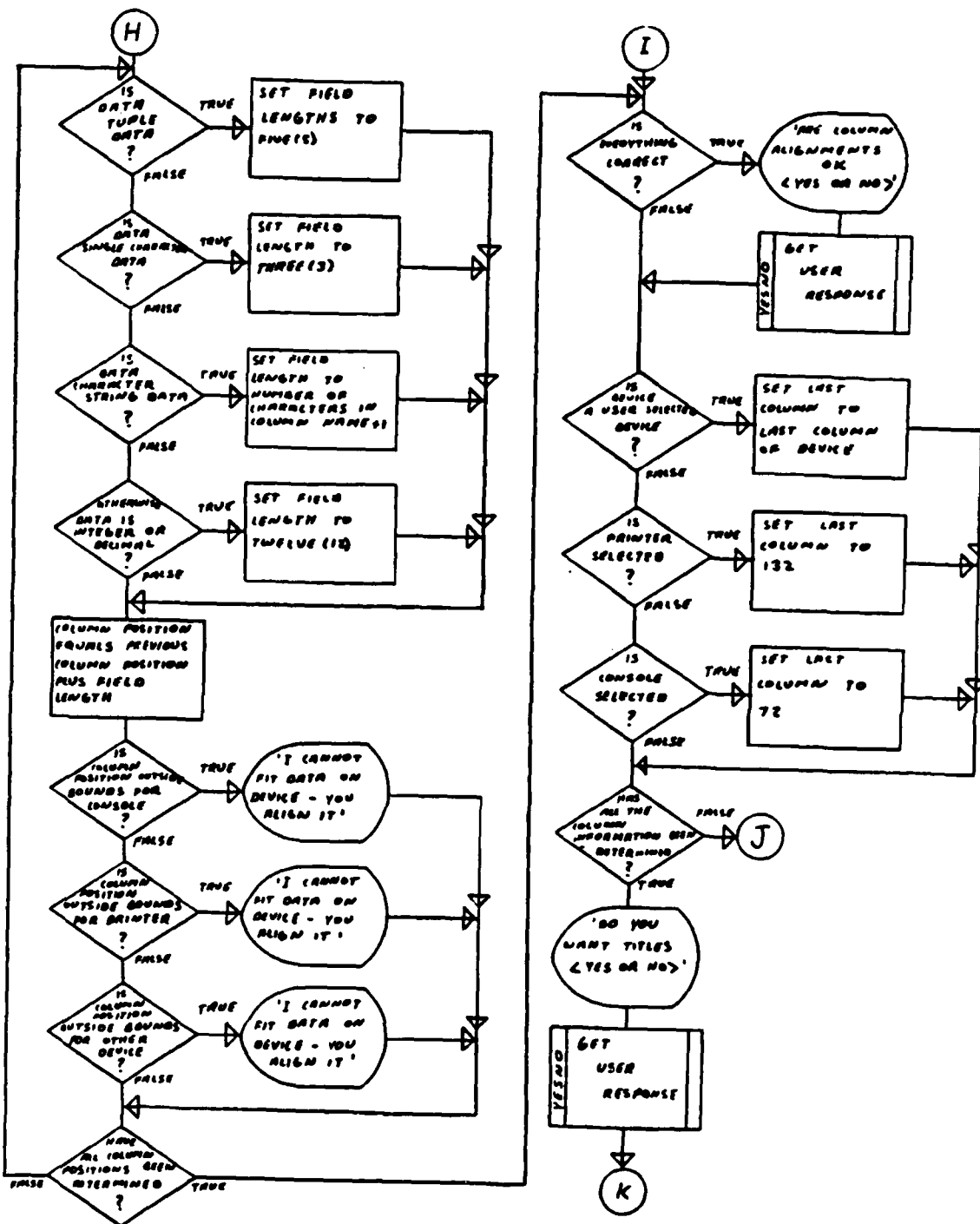


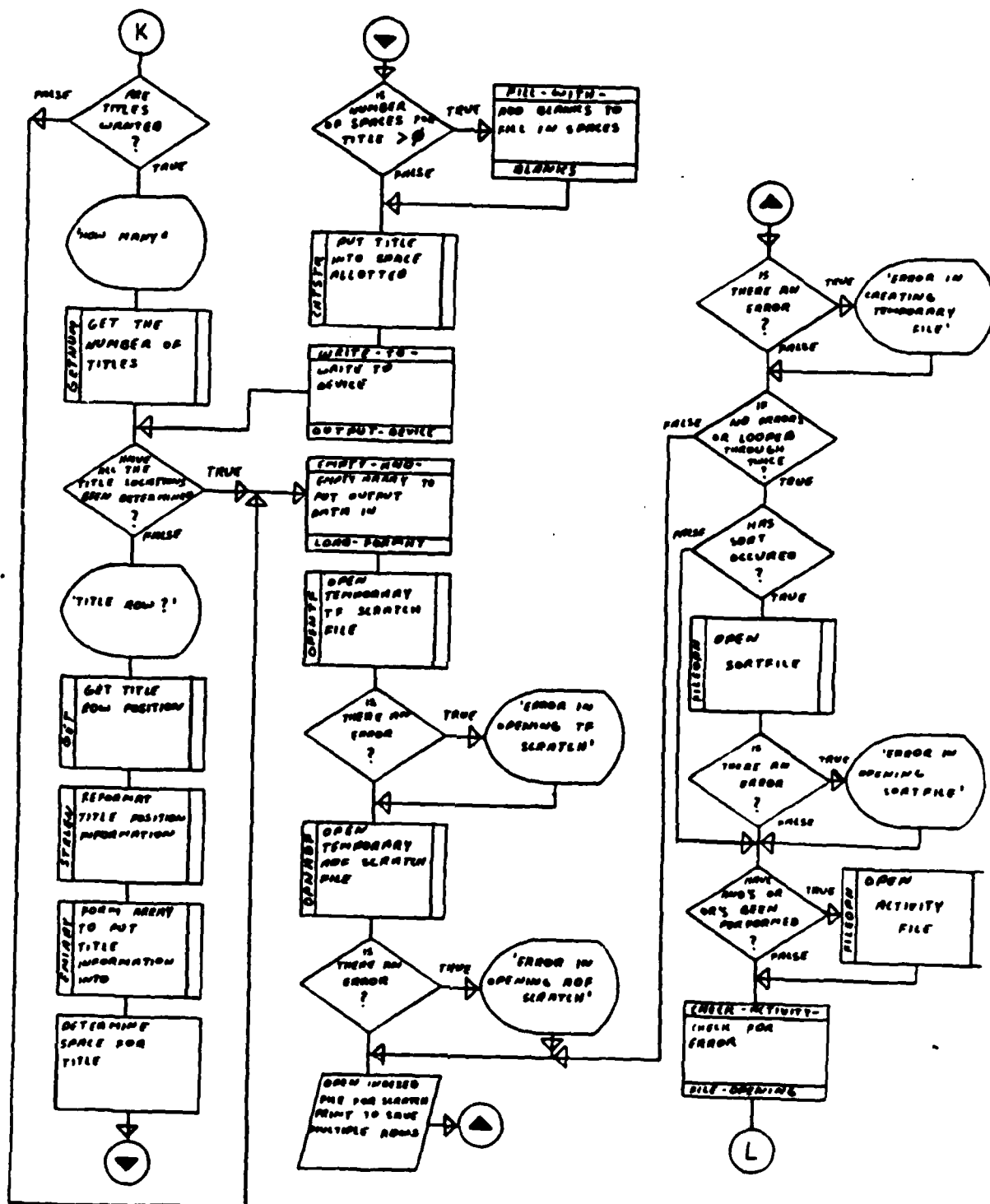




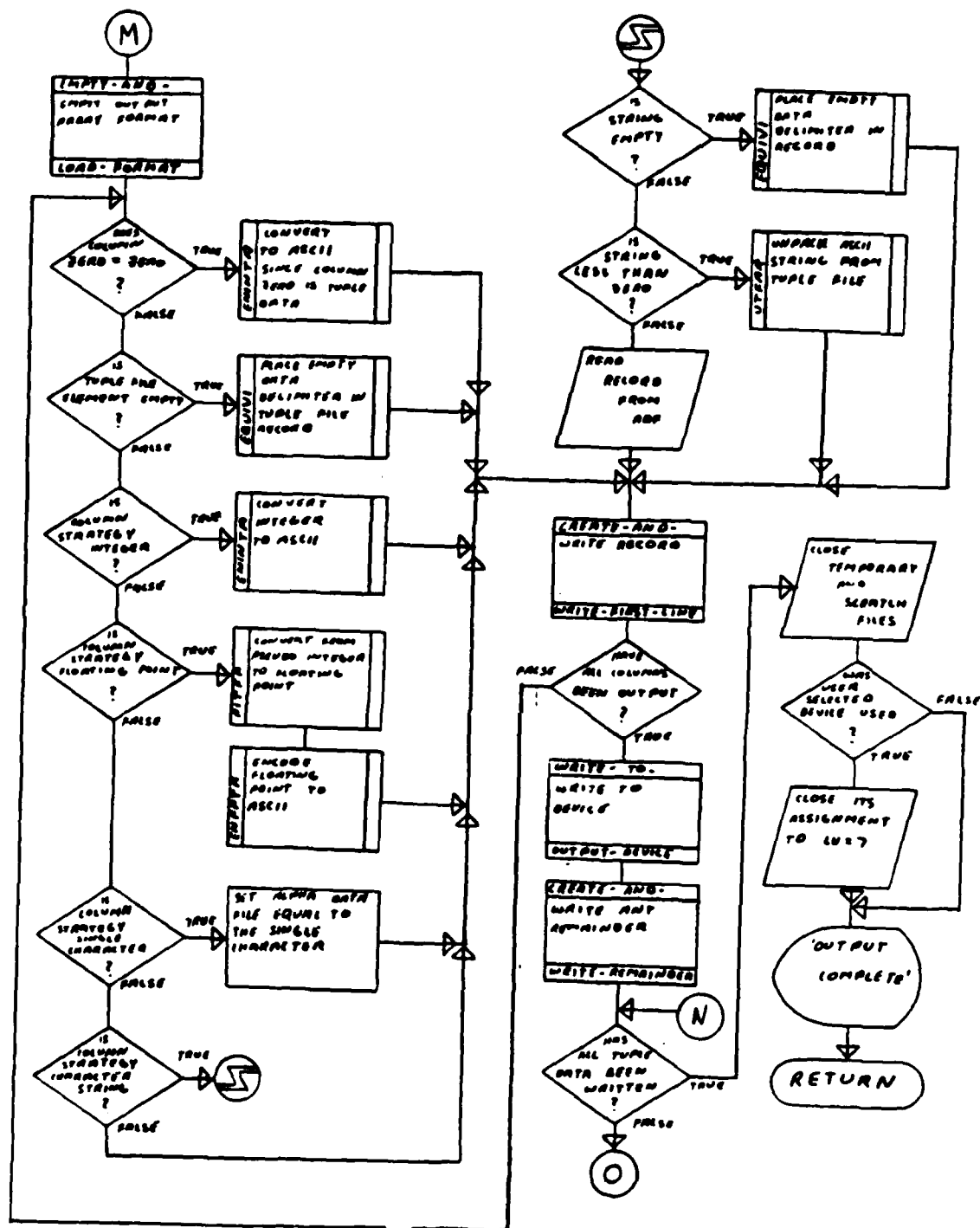


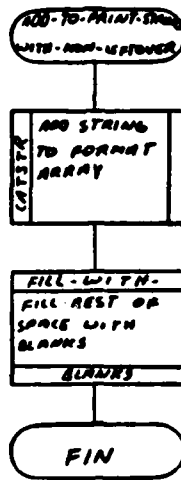


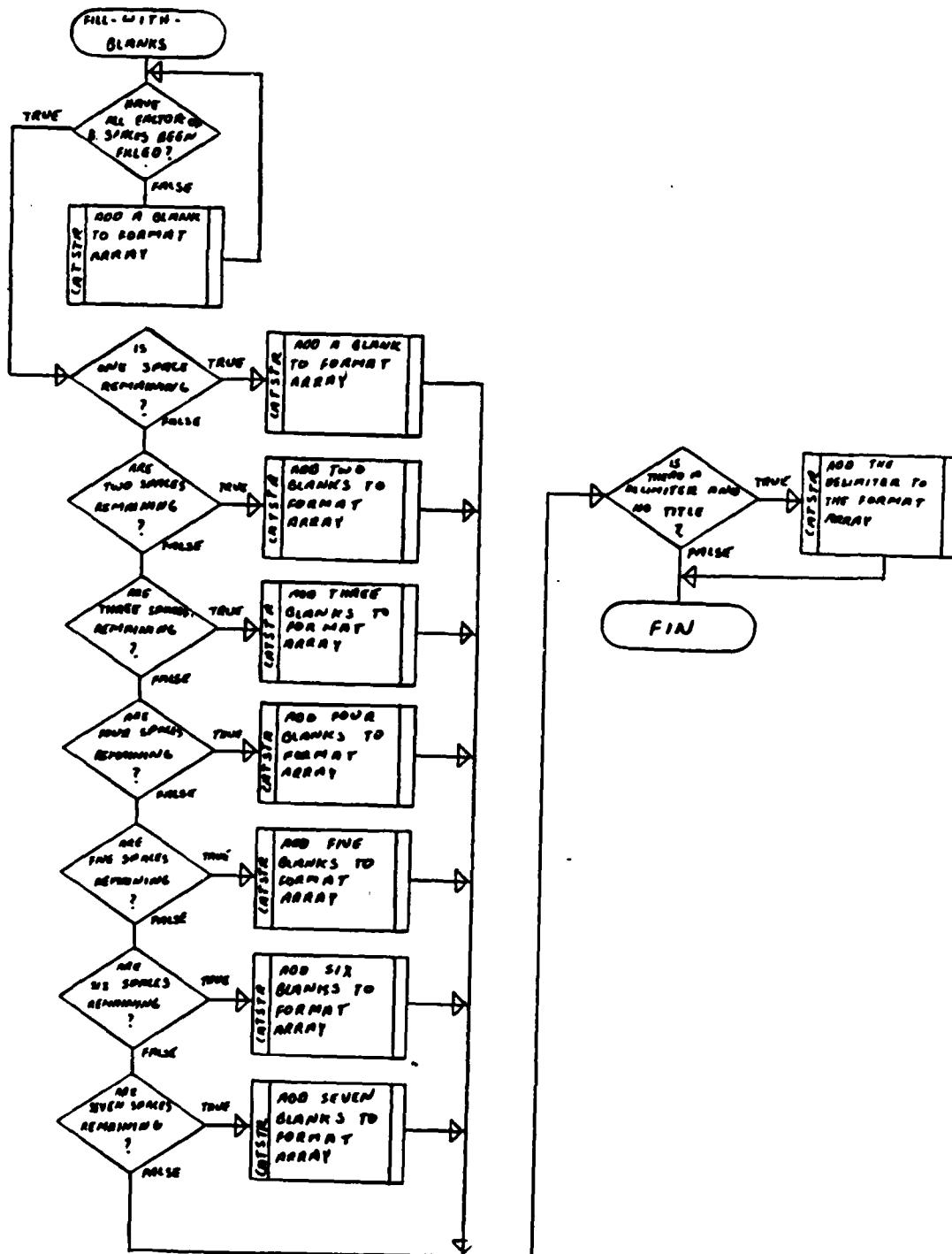


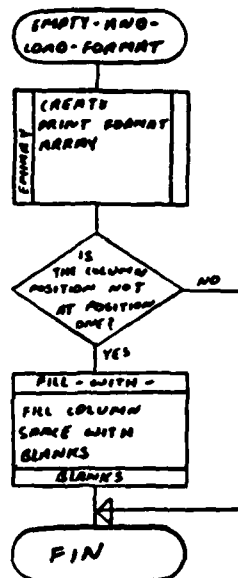




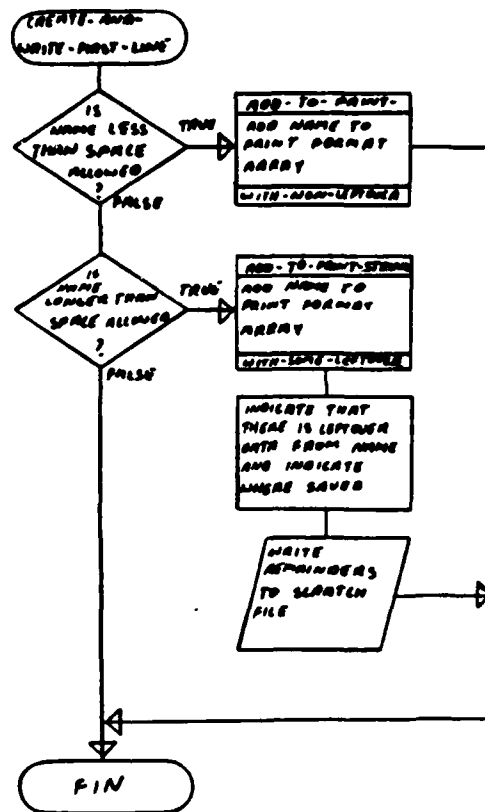


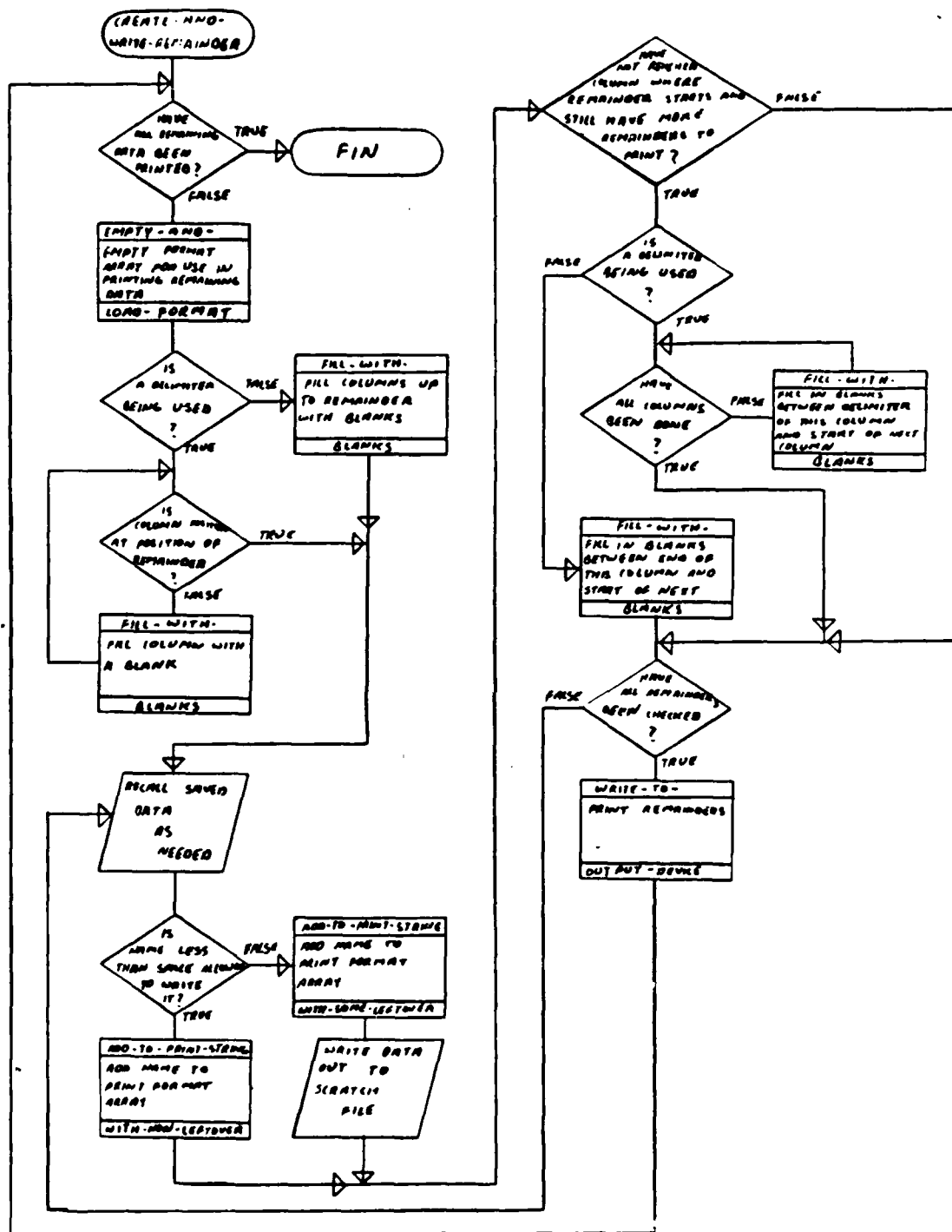


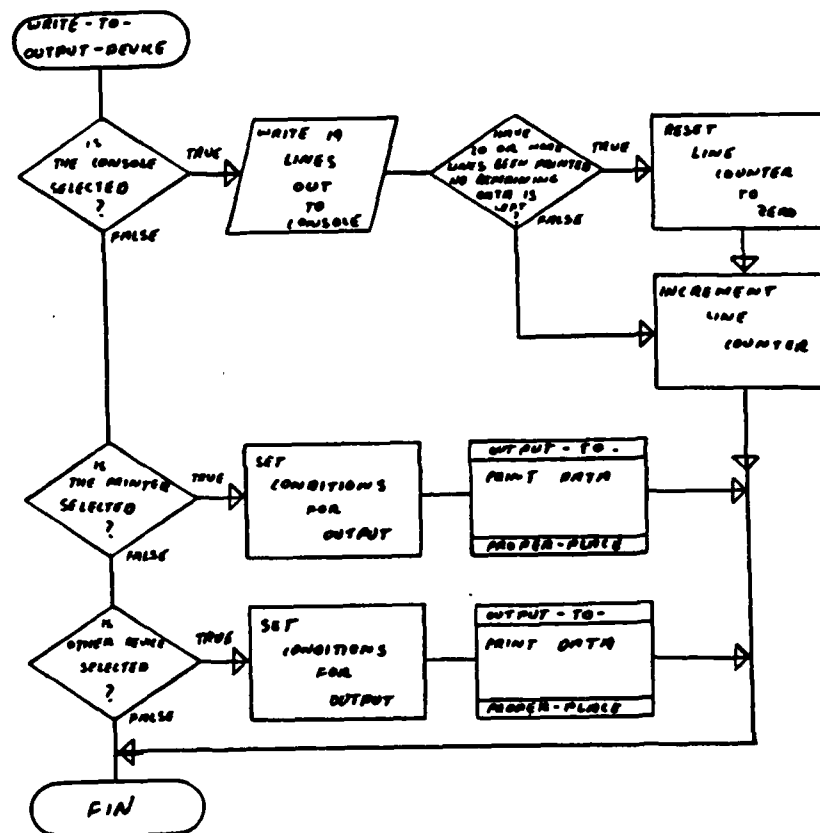




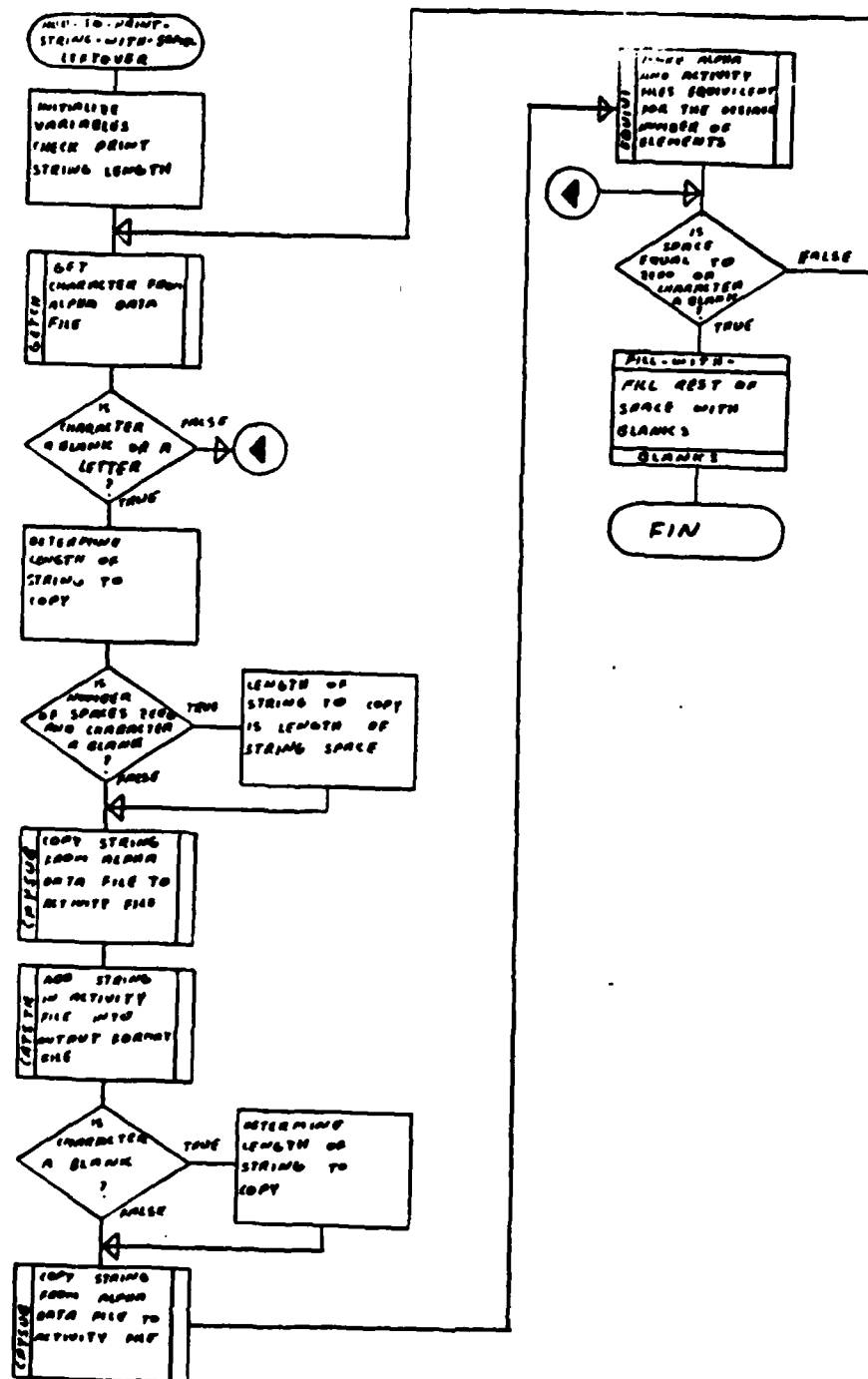


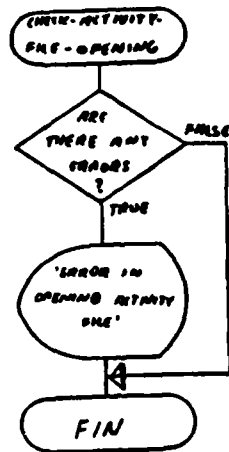


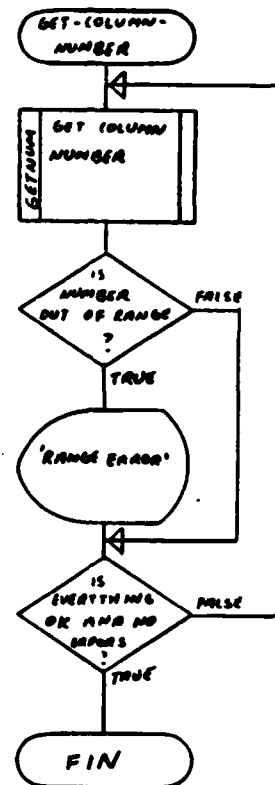












APPENDIX F

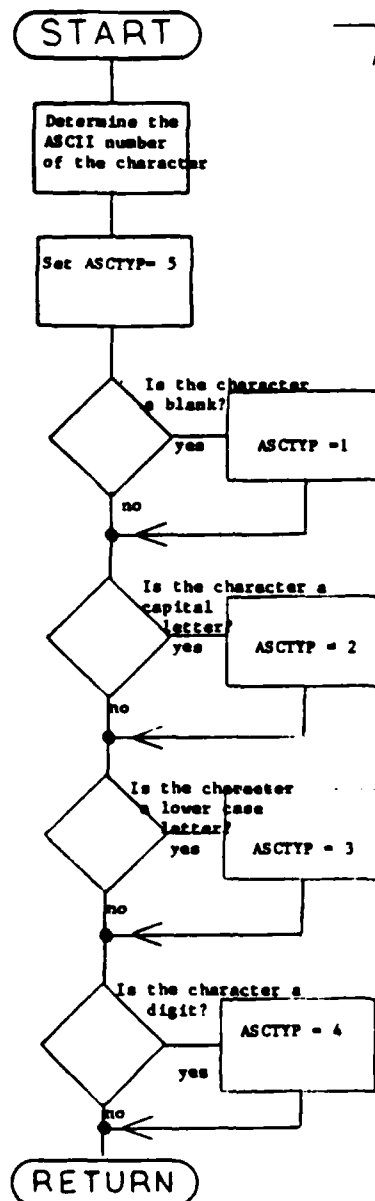
SORT2



Subroutine: SORT

This subroutine allows the user to organize relational data in ascending or decending order. If the relation to be sorted is small (1000 rows or less) then a "fast" sort is done in core. For larger relations, a general file to file weaving bubble sort is implemented.

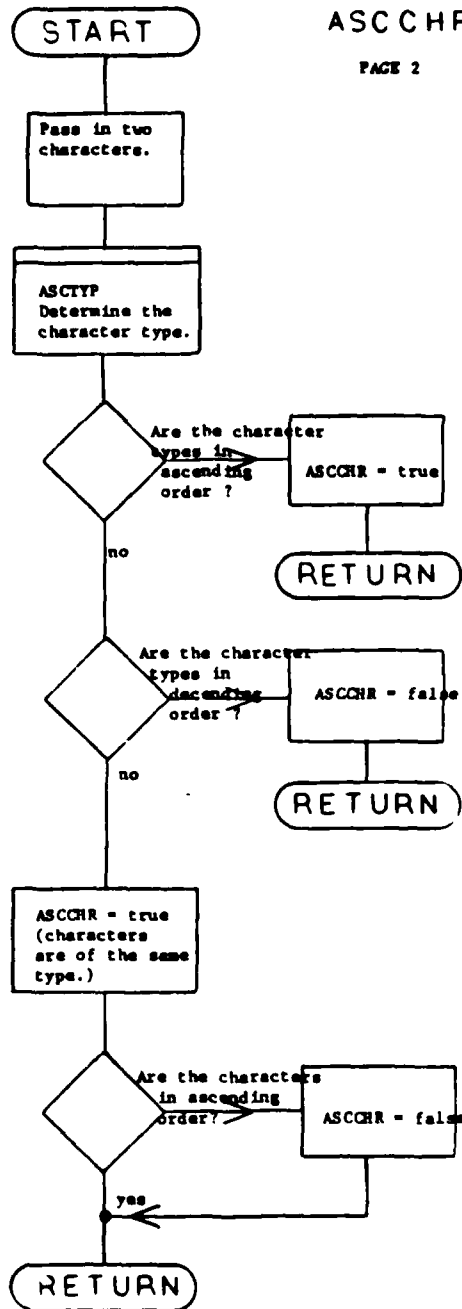
SORT  
ASCTYP  
PAGE 1



# SORT

ASCCHR

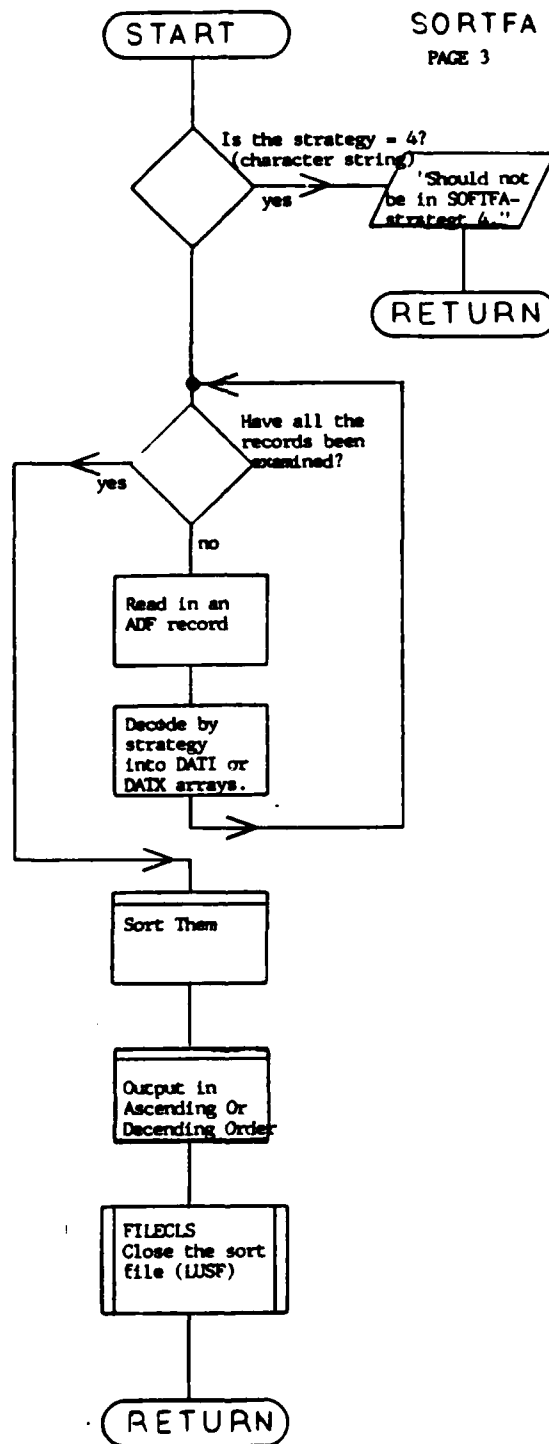
PAGE 2



# SORT

SORTFA

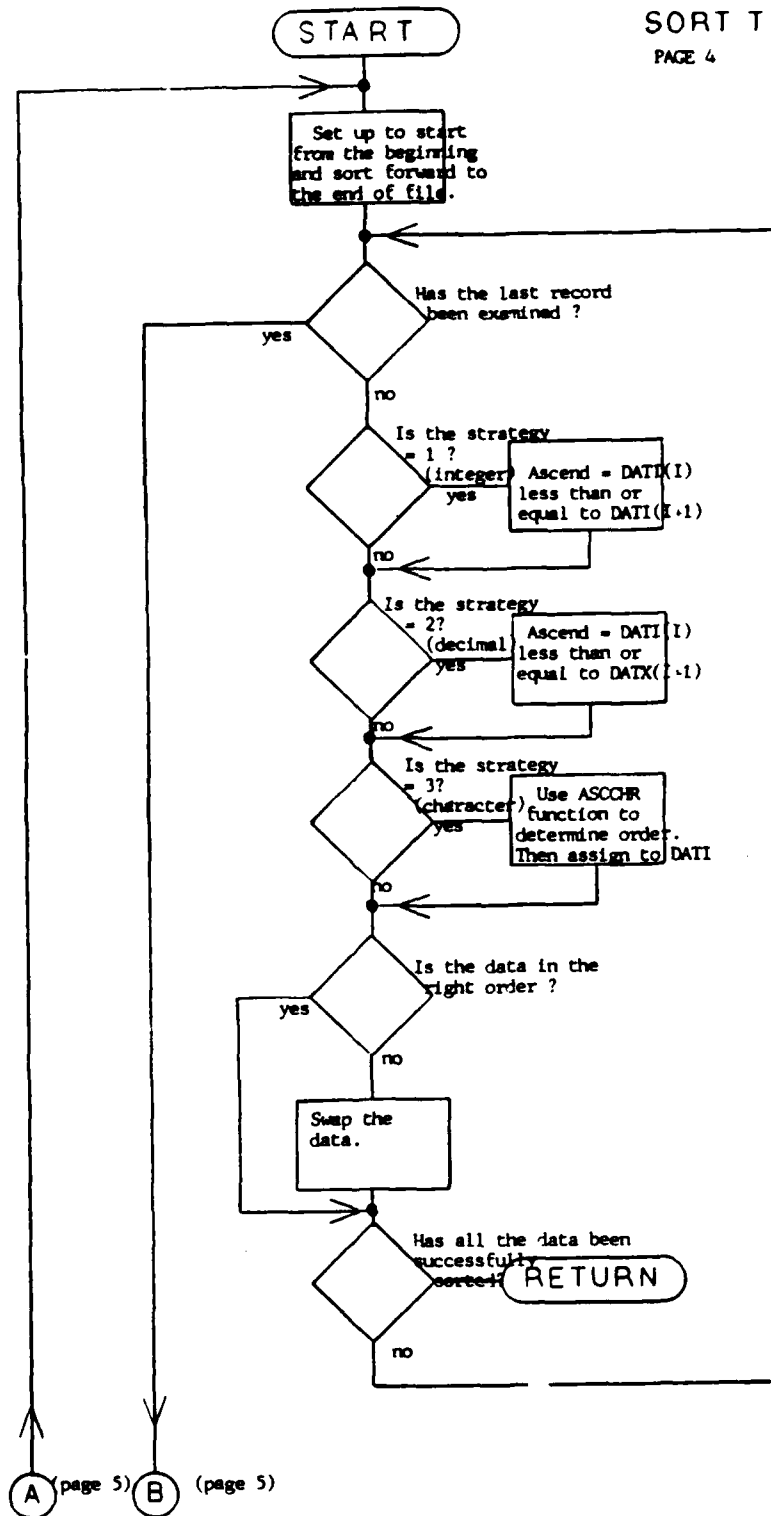
PAGE 3



# SORT

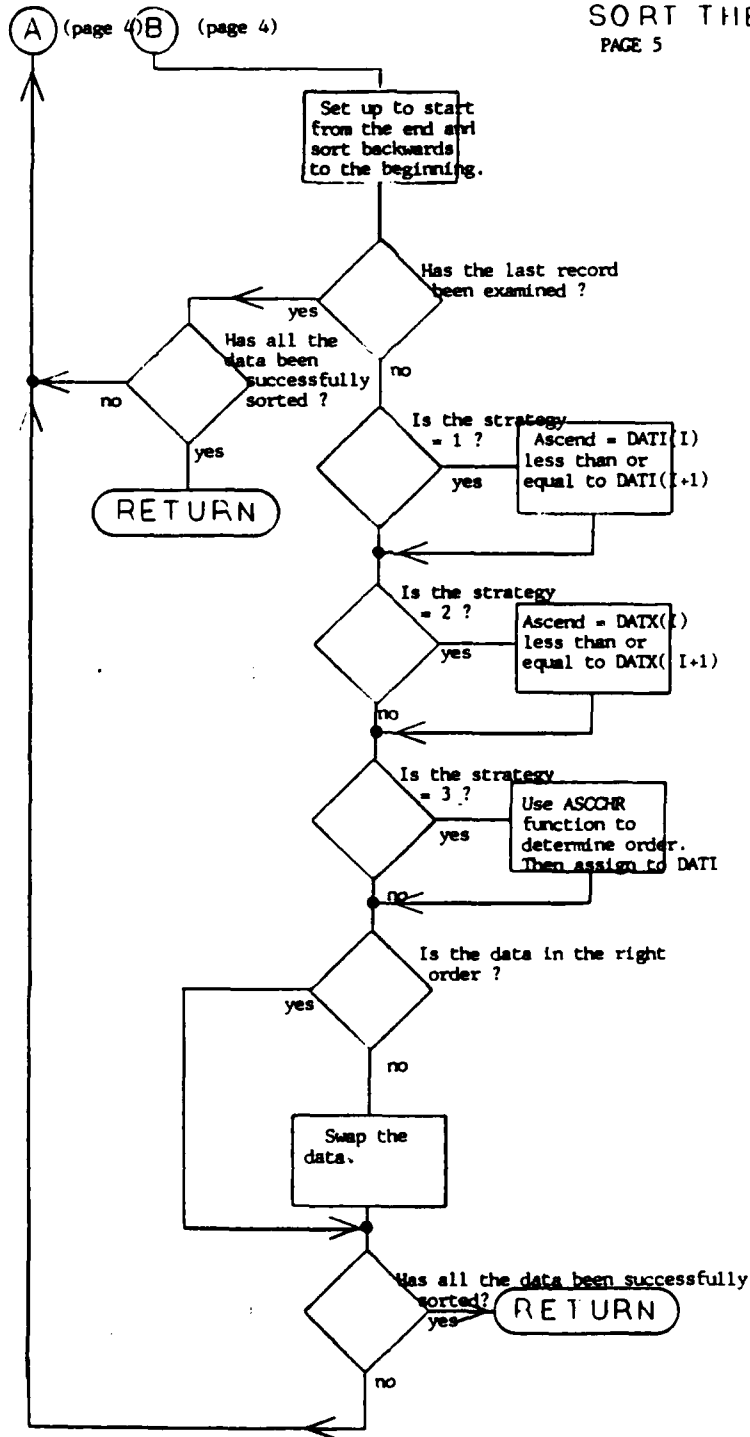
## SORT THEM

PAGE 4



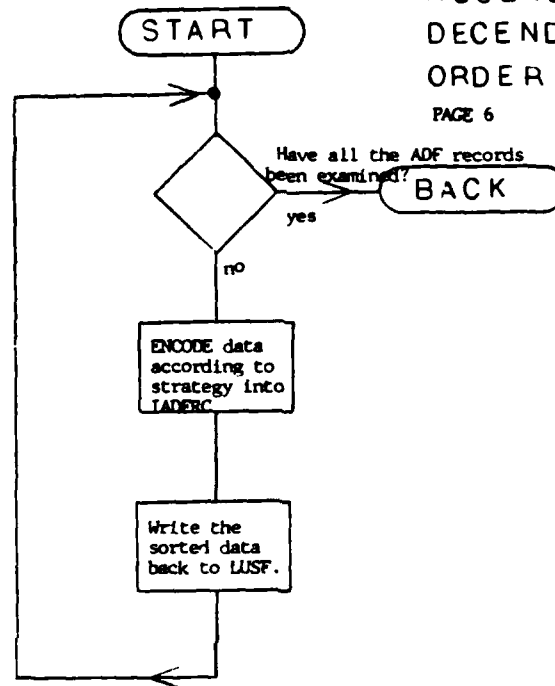
SORT

SORT THEM  
PAGE 5



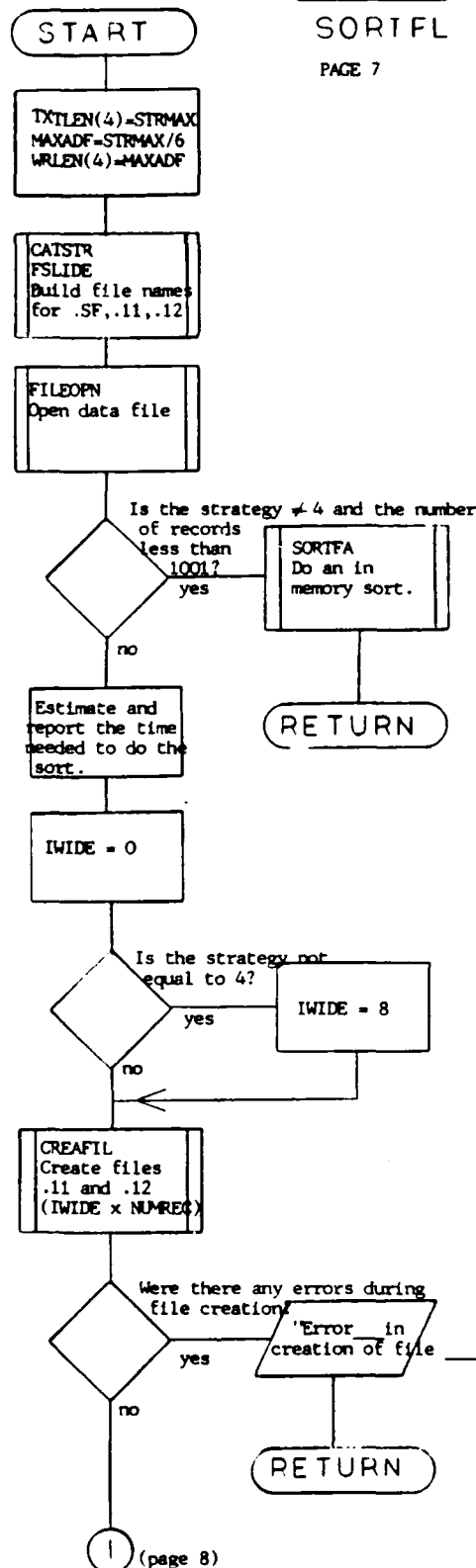
SORT  
OUTPUT IN  
ASCENDING OR  
DECENDING  
ORDER

PAGE 6

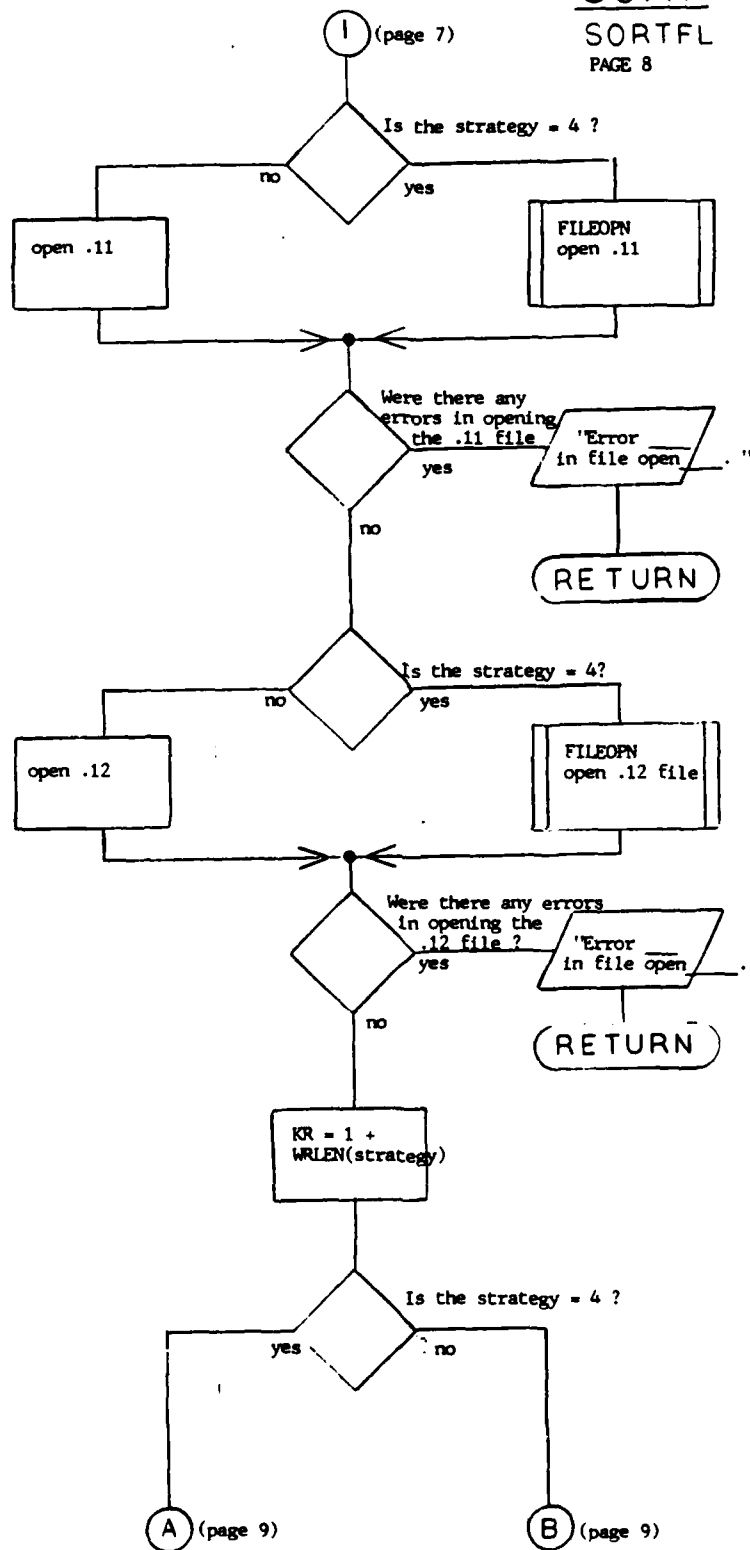


# SORT SORTFL

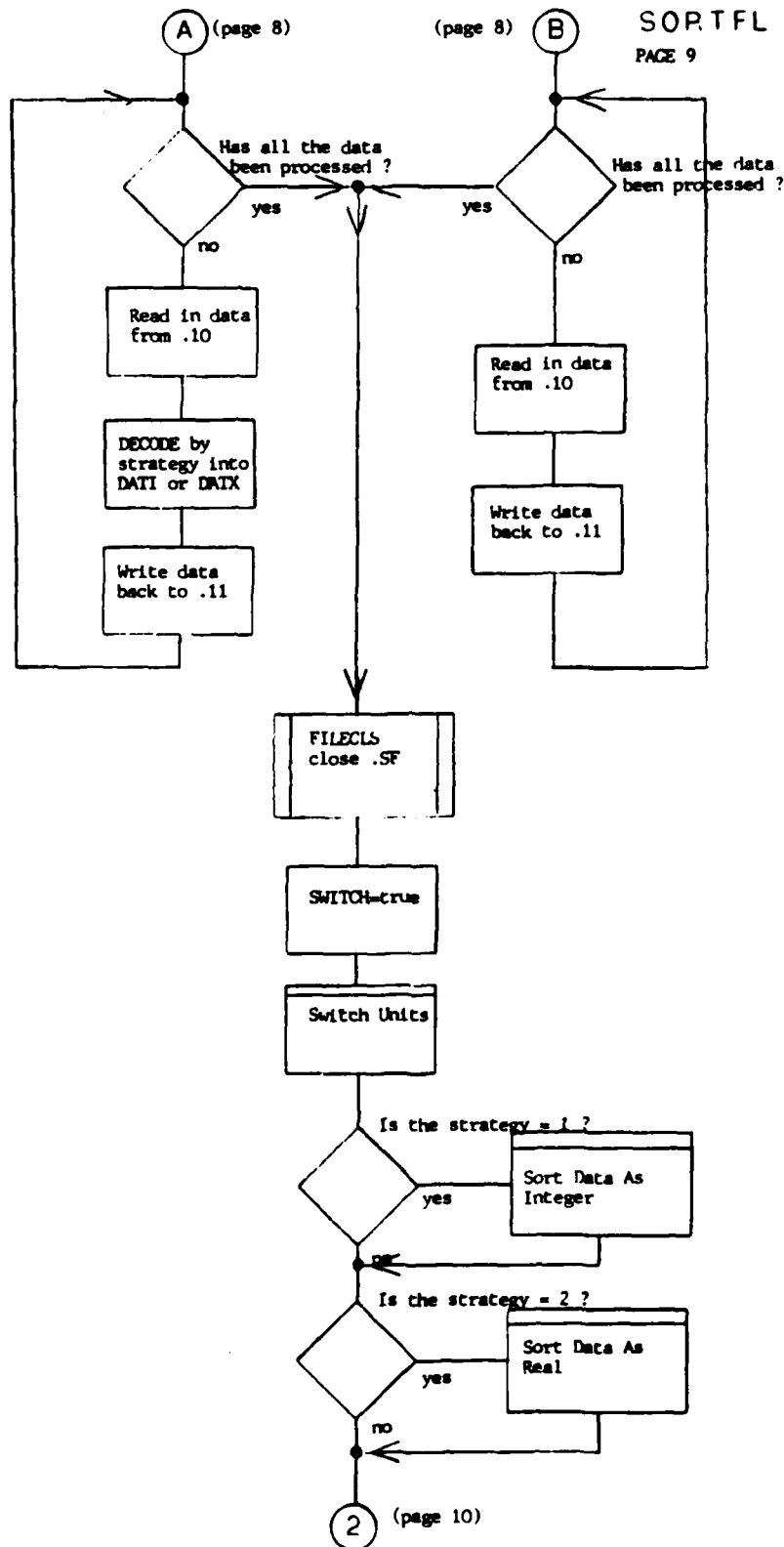
PAGE 7



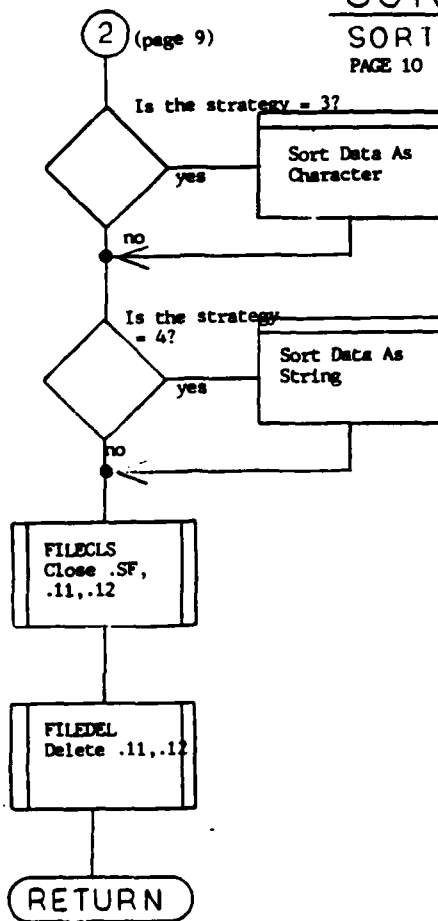




SORT  
SORTFL  
PAGE 9



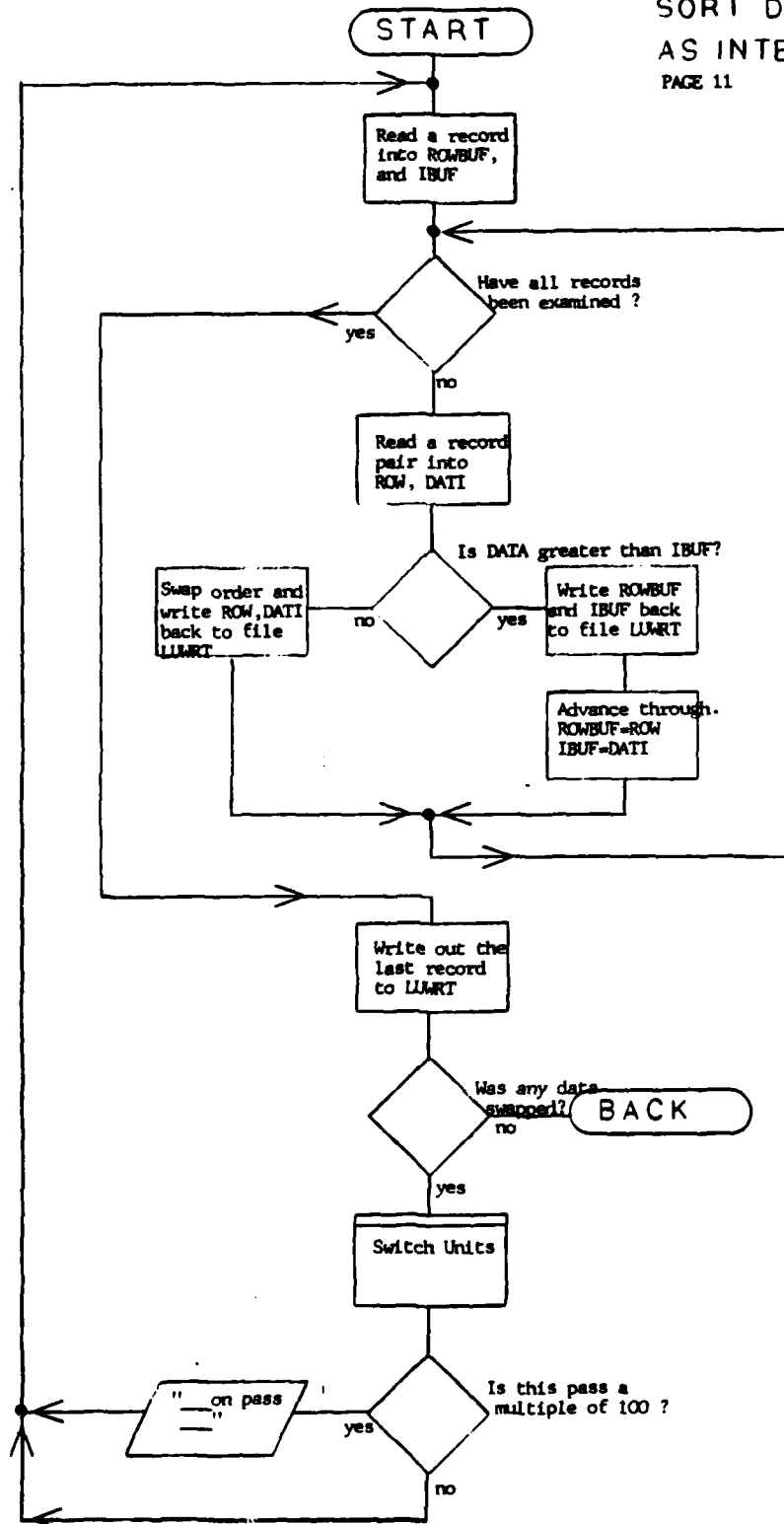
SORT  
SORTFL  
PAGE 10



# SORT

SORT DATA  
AS INTEGER

PAGE 11

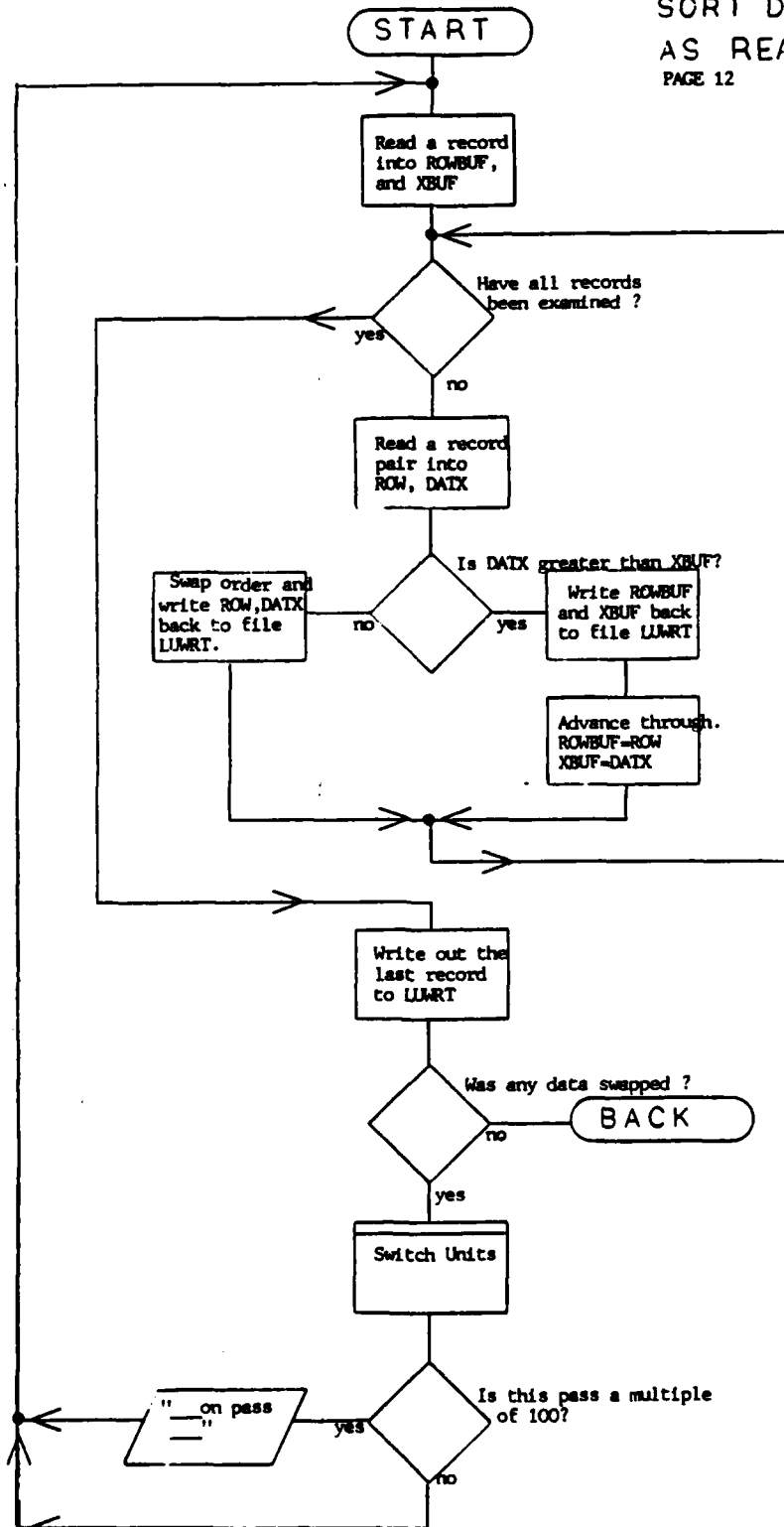


# SORT

SORT DATA

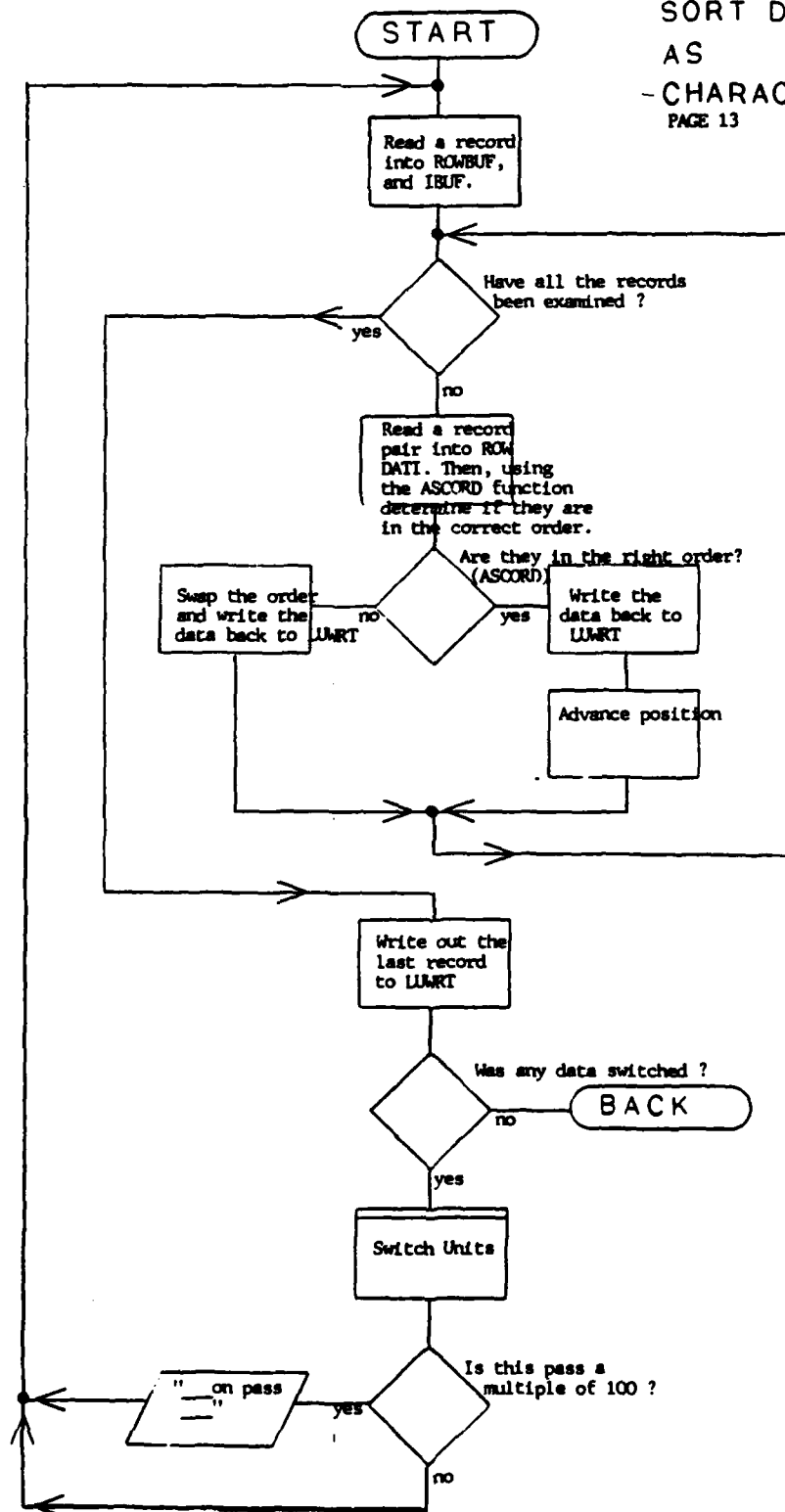
AS REAL

PAGE 12



# SORT

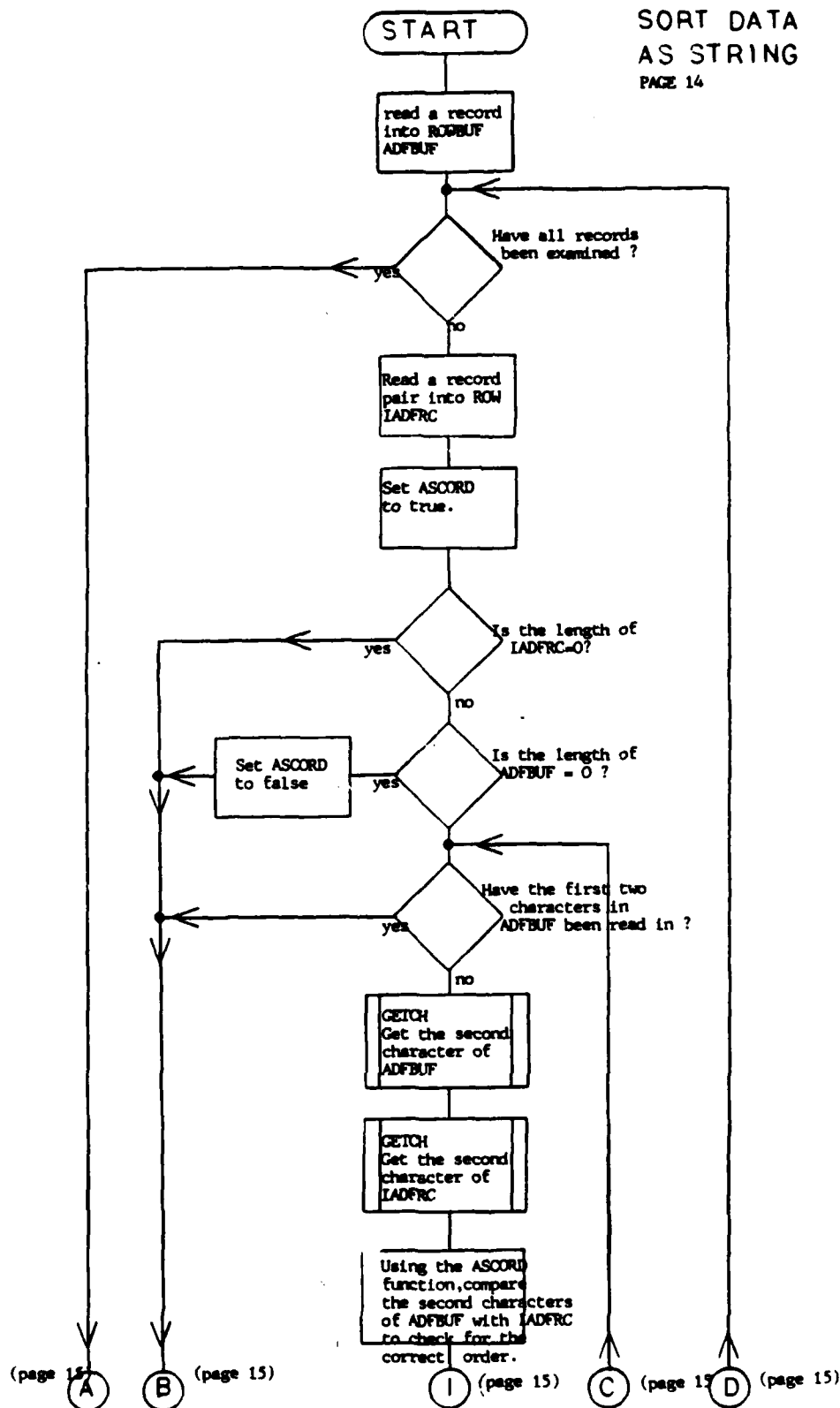
SORT DATA  
AS  
- CHARACTER  
PAGE 13

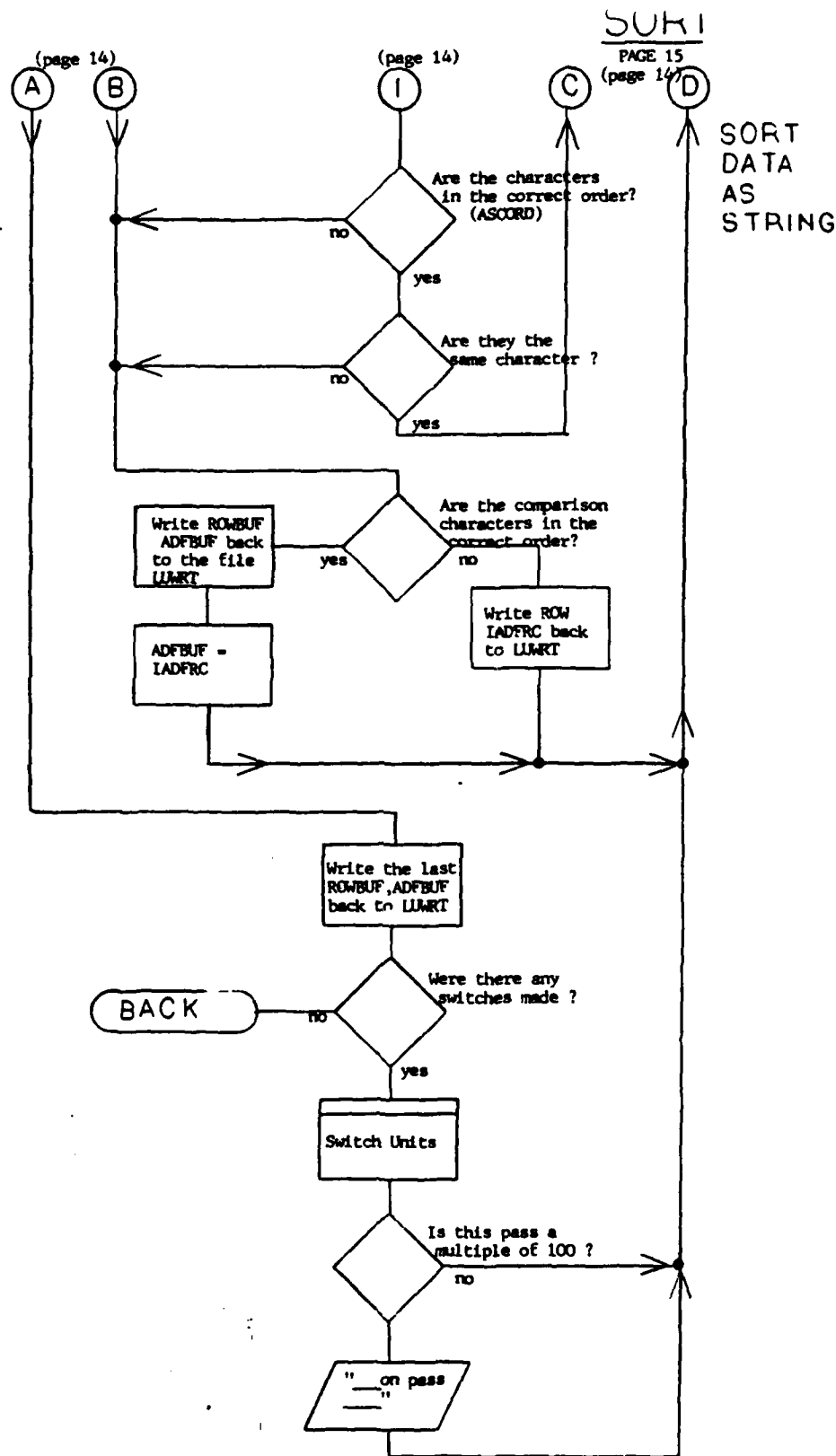


# SORT

SORT DATA  
AS STRING

PAGE 14

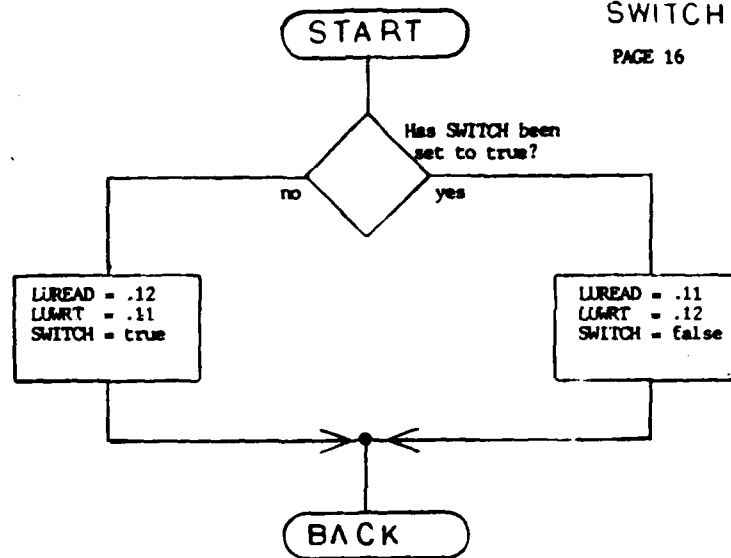


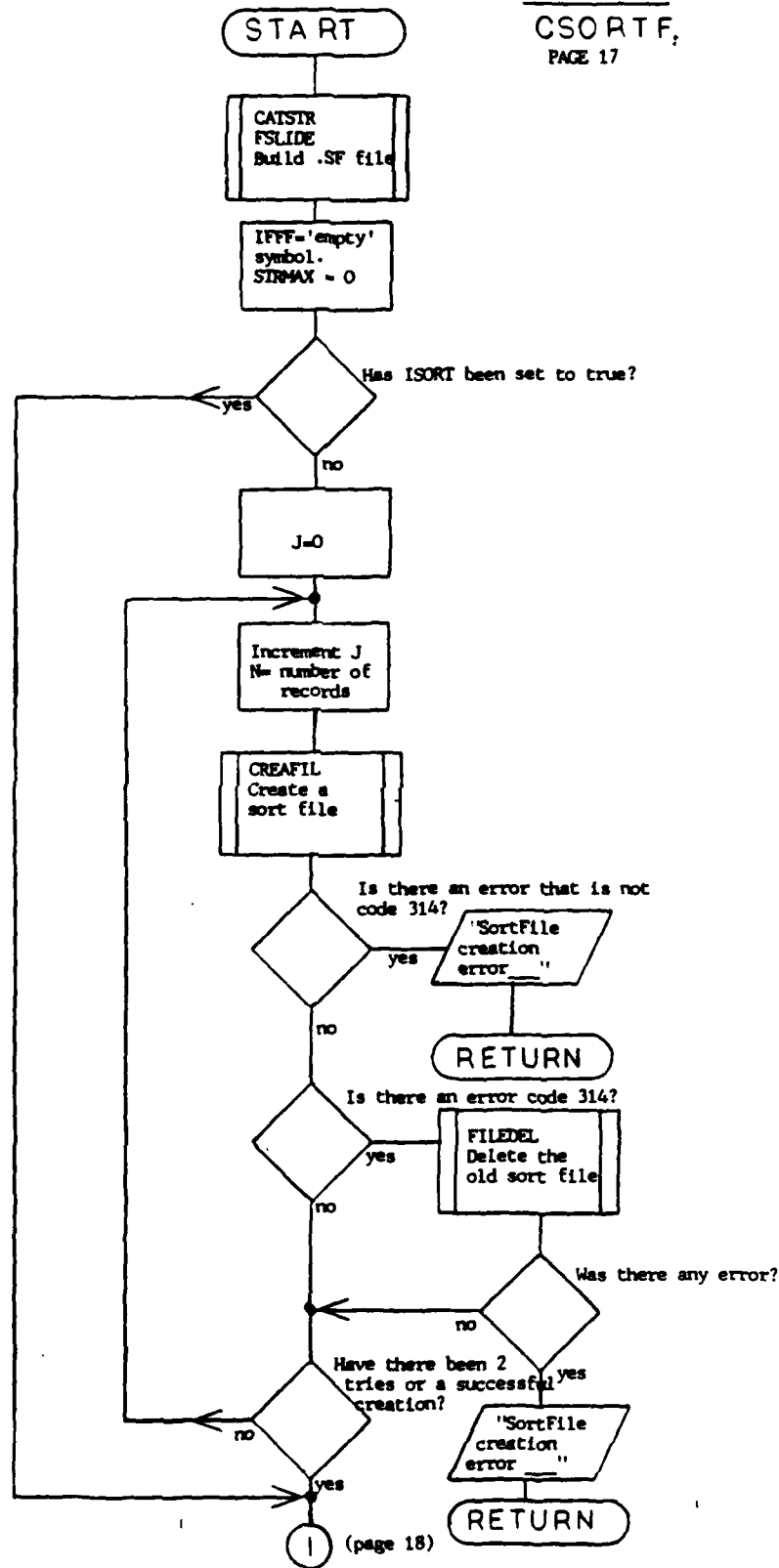




SORT  
SWITCH UNITS

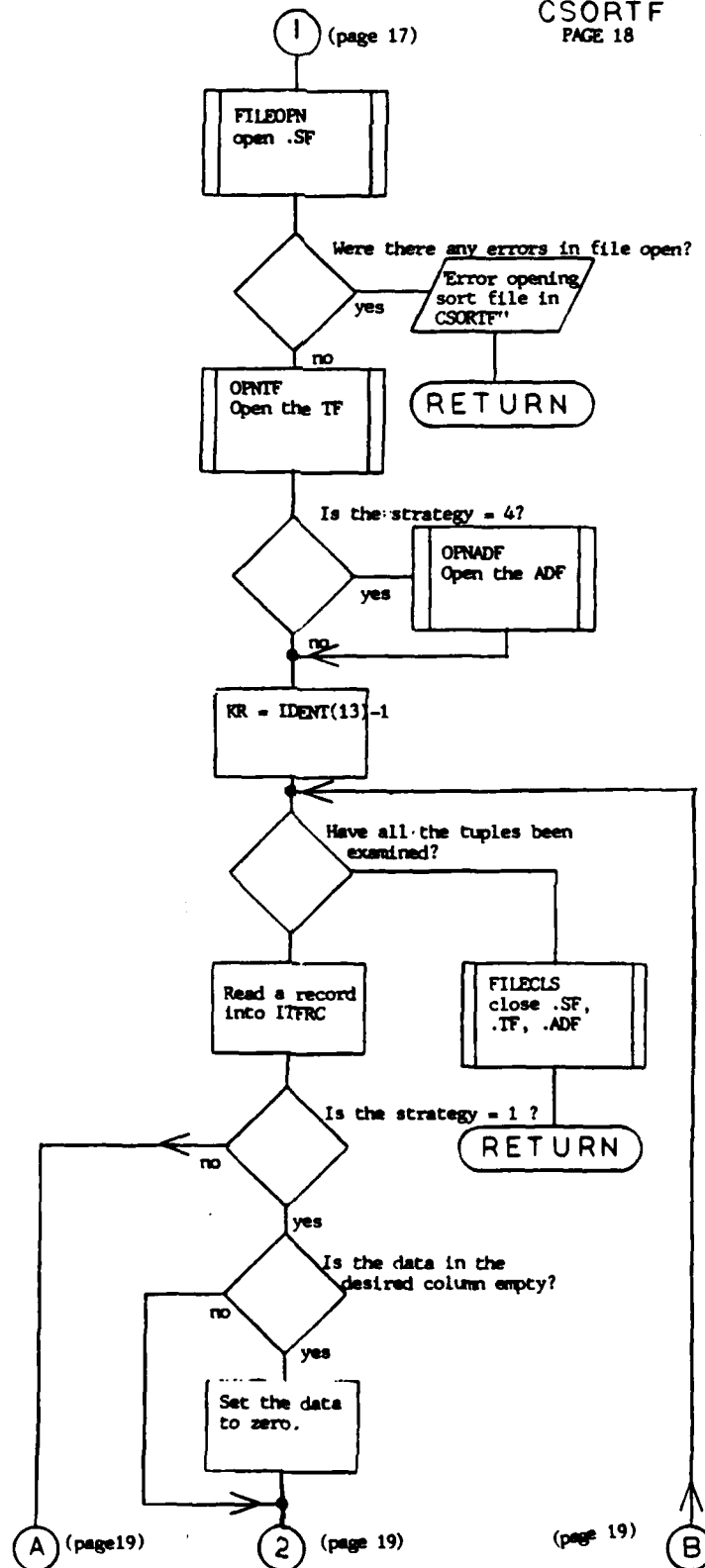
PAGE 16



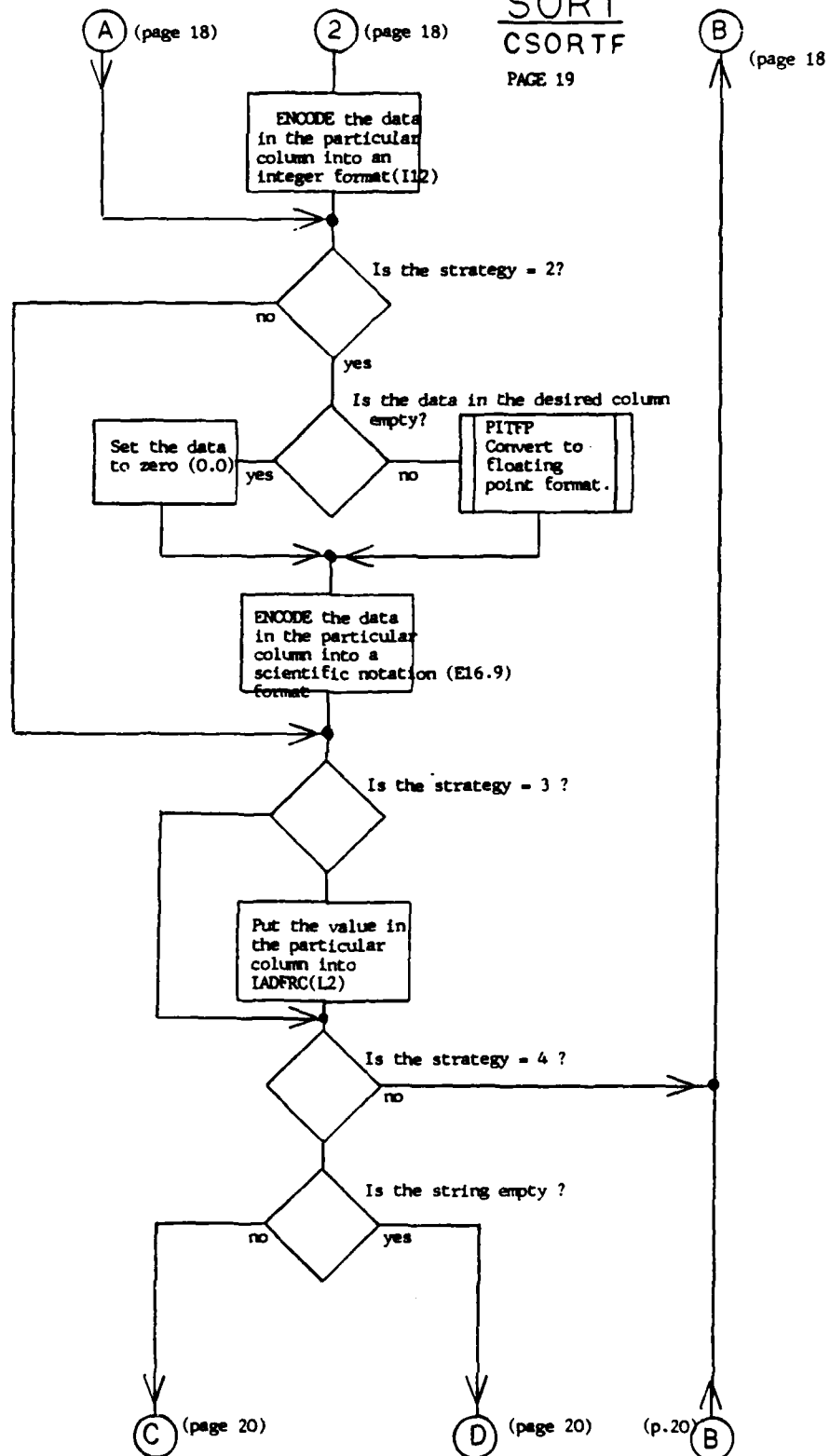


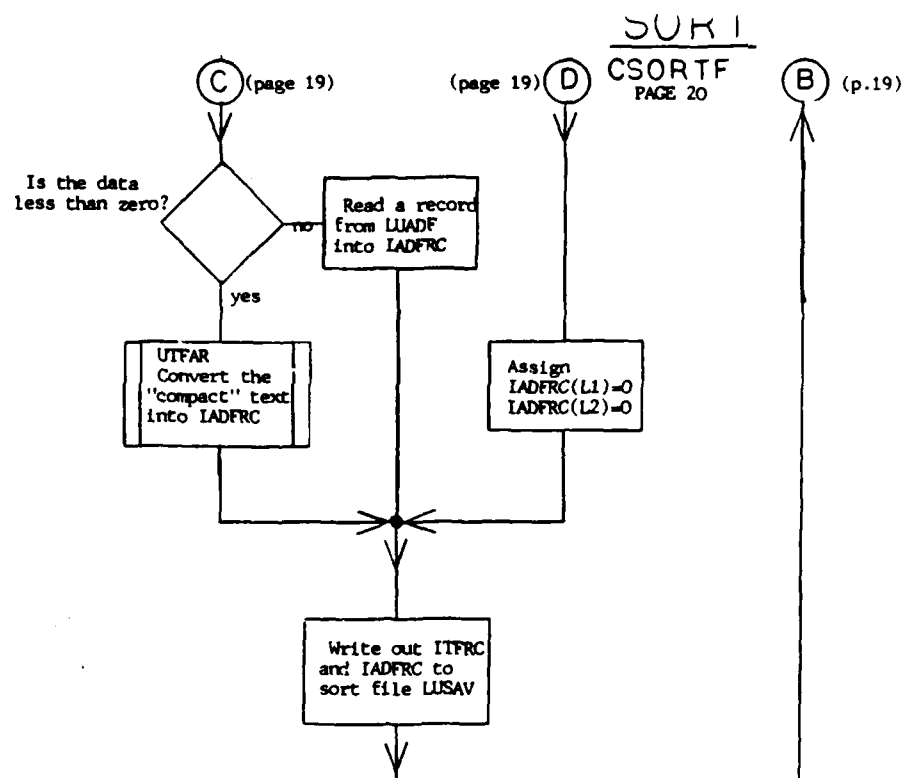
# SORT

CSORTF  
PAGE 18



SORT  
CSORTF  
PAGE 19

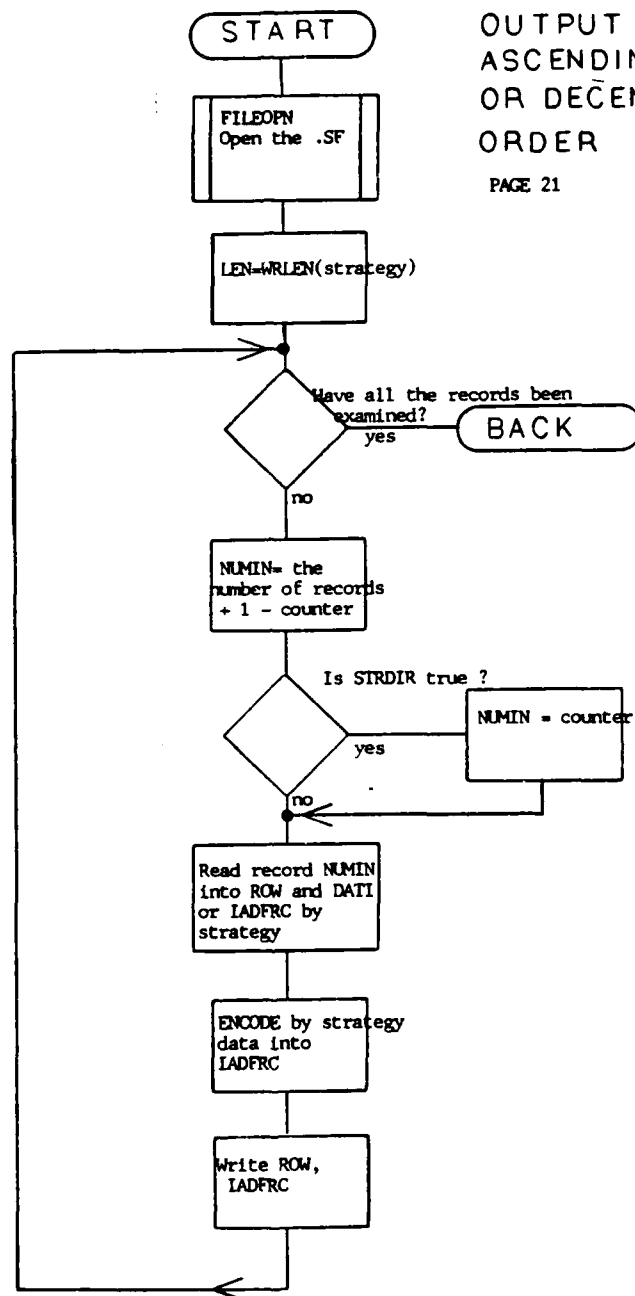




## SORT

OUTPUT IN  
ASCENDING  
OR DECENDING  
ORDER

PAGE 21



APPENDIX G

EDTREL

Subroutine: EDITREL

This subroutine allows the user to change, add, delete, or examine tuple data.

The following edit commands are available to the user:

<COMMAND>	<FUNCTION>
?(STATUS)	GIVES CURRENT VALUES OF ROW AND COLUMN POINTERS, RELATION NAME, NUMBER OF RECORDS AND NUMBER OF COLUMNS
Q(QUIT)	TERMINATES EDITING
C(COLUMNS)	LIST NAMES OF EACH COLUMN BY NUMBER IN THE RELATION
+[] (PLUS)	MOVES CURRENT ROW POINTER FORWARD [X] ROWS (LAST GIVEN IF TOTAL EXCEEDED) 1 IS UNDERSTOOD IF [0] OR [ ] USED.
-[] (MINUS)	SAME AS + EXCEPT IN OTHER DIRECTION
F(FIND)	FIND THE FIRST OCCURANCE OF DATA ITEM IN SPECIFIED COLUMN.
S(SUBSTITUTE)	USER SUBSTITUTES NEW DATA INTO LOCATION POINTED TO BY CURRENT ROW AND CURRENT COLUMN POINTERS.
E(EXAMINE)	DISPLAYS TO USER CONTENTS OF CURRENT ROW AND COLUMN POINTER LOCATION.
D[] (DELETE)	DELETES [X] NUMBER OF TUPLES STARTING WITH CURRENT TUPLE. NONE DELETED IF NONE SPECIFIED.
I(INSERT)	INSERTS TUPLE AFTER CURRENT TUPLE POINTER. PROMPTS USER FOR COLUMN VALUES.
B(BOTTOM)	SAME AS INSERT ACCEPT PLACES TUPLE AT BOTTOM.



P[] (PRINT)	PRINTS [X] TUPLES OF RELATION STARTING WITH OUTPUT DATA IN A FORMAT THAT WILL ALLOW RE'ADD'ING BY THAT OR ANOTHER RELATION WITH EQUAL STRATEGIES WITH CURRENT TUPLE. 1 ASSUMED IF 0 OR BLANK.
R(RE-START)	USER MAY RESTART EDIT PROCESS TO CHANGE DISPLAY FORMAT.
H(HELP)	LIST THE POSSIBLE COMMANDS
A(ADD TUPLES)	READS DATA TUPLES INTO RELATION IN FREE FORMAT,CARD IMAGE SEPARATED BY USER DELIMITER.

FEASIL  
EDITREL  
Subroutines and Procedures

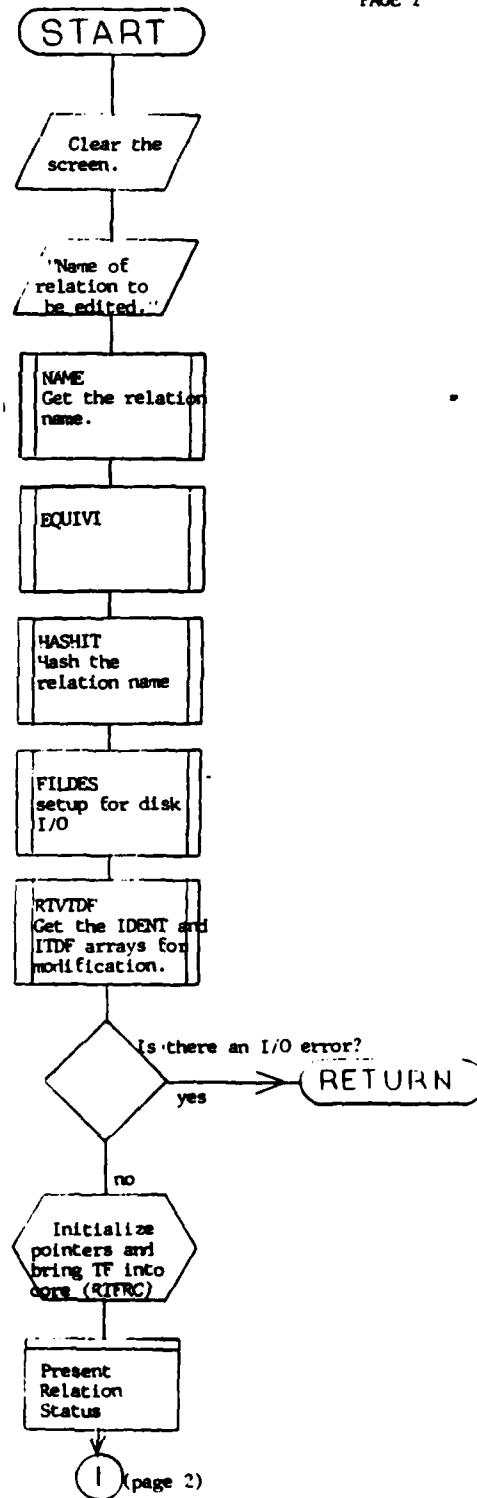
(Listed in the order in which they appear in the code.)

Subroutine or Procedure Name.....	Flow Chart	Page number
I Quit .....		Page 7
Display Help .....		Page 7
Check If Relation Empty .....		Page 8
Present Relation Status .....		Page 8
Find Column Display Description .....		Page 9
Insert A Tuple.....		Page 11
Delete Tuples.....		Page 12
Add Tuples At Bottom .....		Page 13
Print Data .....		Page 14
Print Variable Length Output .....		Page 15
Generate Write Statement Maybe Column Names ....		Page 16
Add Tuple Data To Block .....		Page 17
Restart Description .....		Page 18
Move Tuple Pointer Forward .....		Page 19
Get Next Tuple .....		Page 20
Move Tuple Pointer Backwards .....		Page 20
Find Value .....		Page 21
Substitute New Value .....		Page 25
Examine Current Values .....		Page 26
Idle and Do Nothing .....		Page 26
Present Current Value .....		Page 27
Get Column Number .....		Page 27
Get Value or String .....		Page 28
Insert Tuple Data .....		Page 32
Display Selected Tuple Values.....		Page 34
Check TF Allocation .....		Page 34
Add Data Tuples To Relation .....		Page 35
Process All Cards .....		Page 40
Input As Non String .....		Page 43
Copy String Between Delimiter.....		Page 46

Find The next Delimiter .....	Page 46
Read The Next Data Card .....	Page 47
Write Data Card Read Error .....	Page 48
Write Data File End of File .....	Page 48
Assign To Longest Displayed Column Name .....	Page 48
Output Tuple Data .....	Page 49
Get Column Names Into Core .....	Page 53
List Column Names .....	Page 54
Blank Format Array .....	Page 54
Save Like Add Command Expects .....	Page 55
Display prompt .....	Page 59
Format Decimal ASCII Array By Mag .....	Page 59

# EDITREL

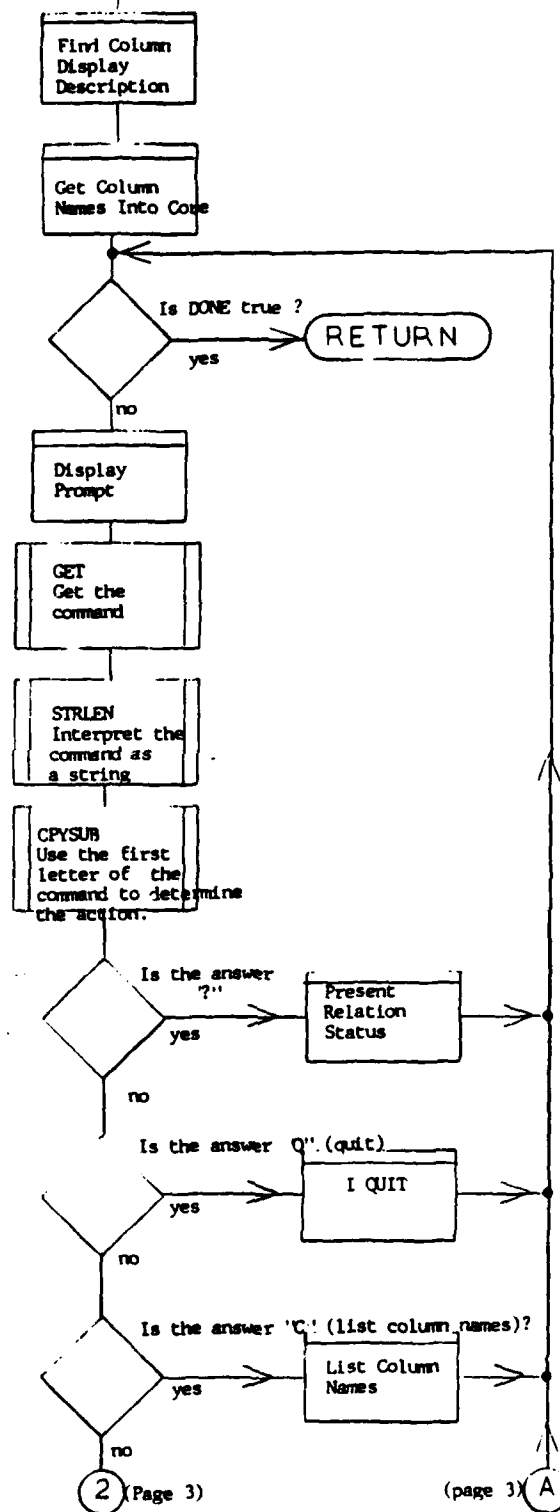
PAGE 1



① (page 1)

EDITH-EL

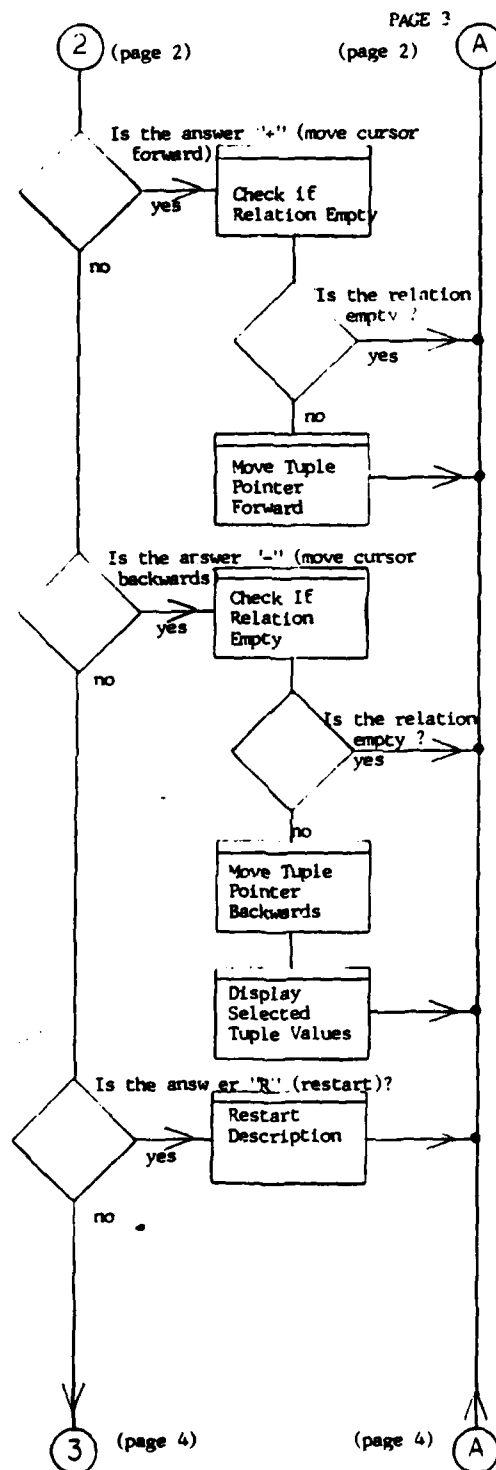
PAGE 2

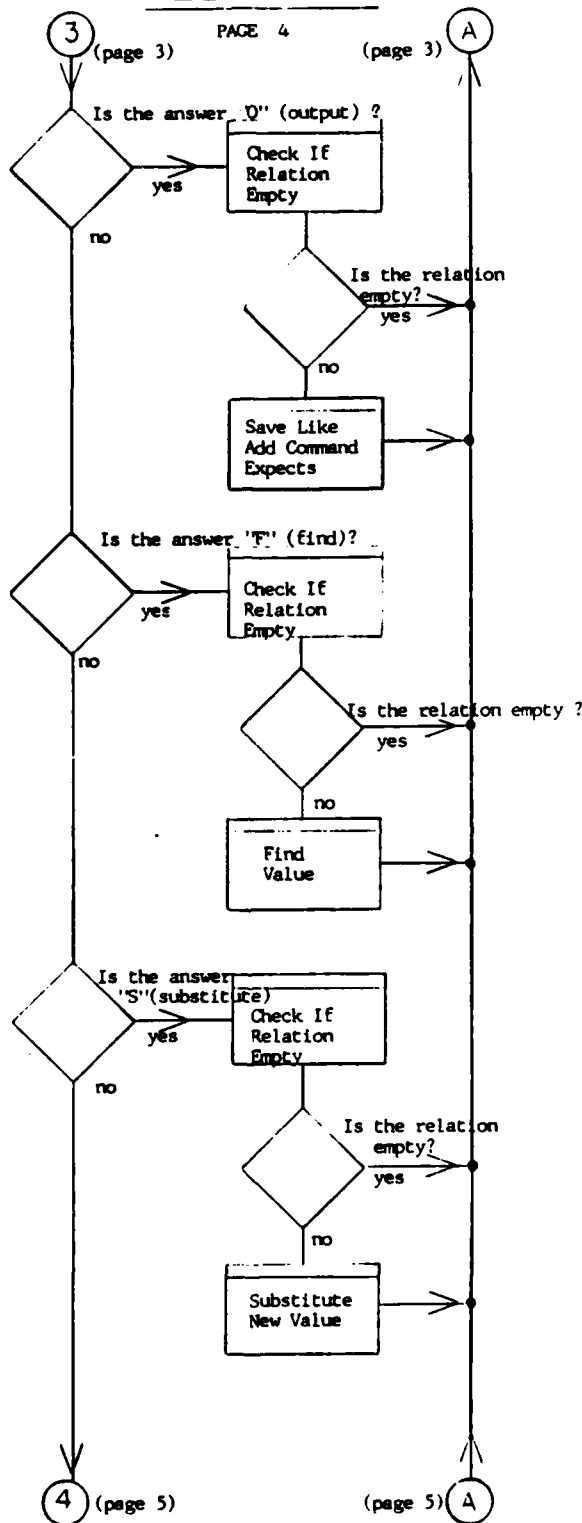


② (Page 3)

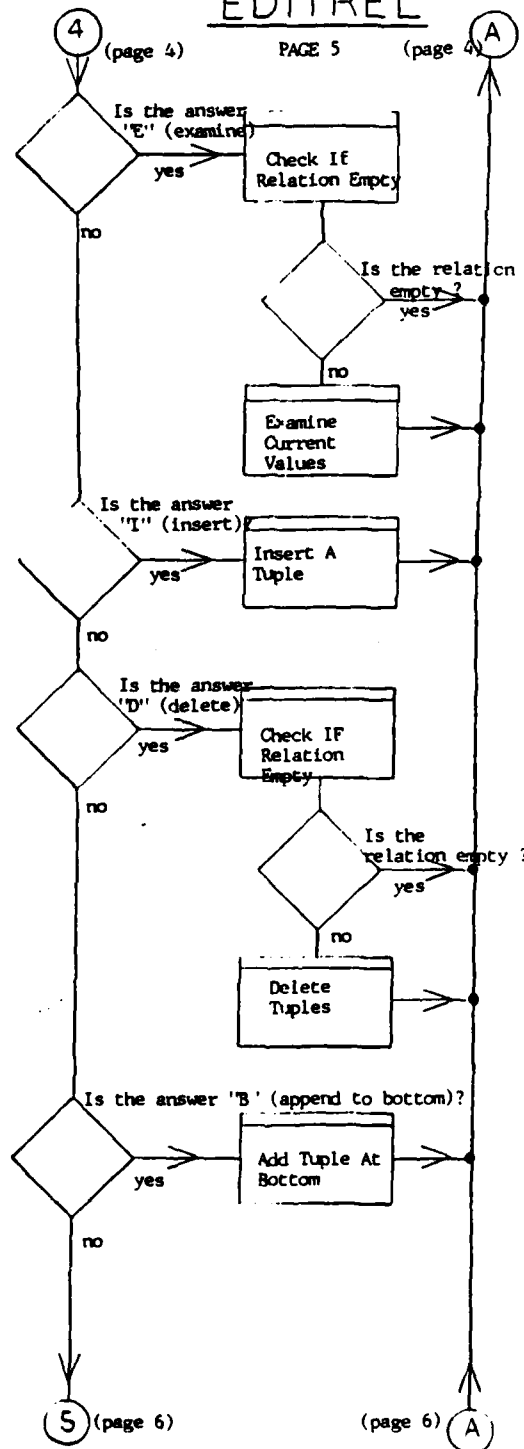
(page 3) A

# EDITREL



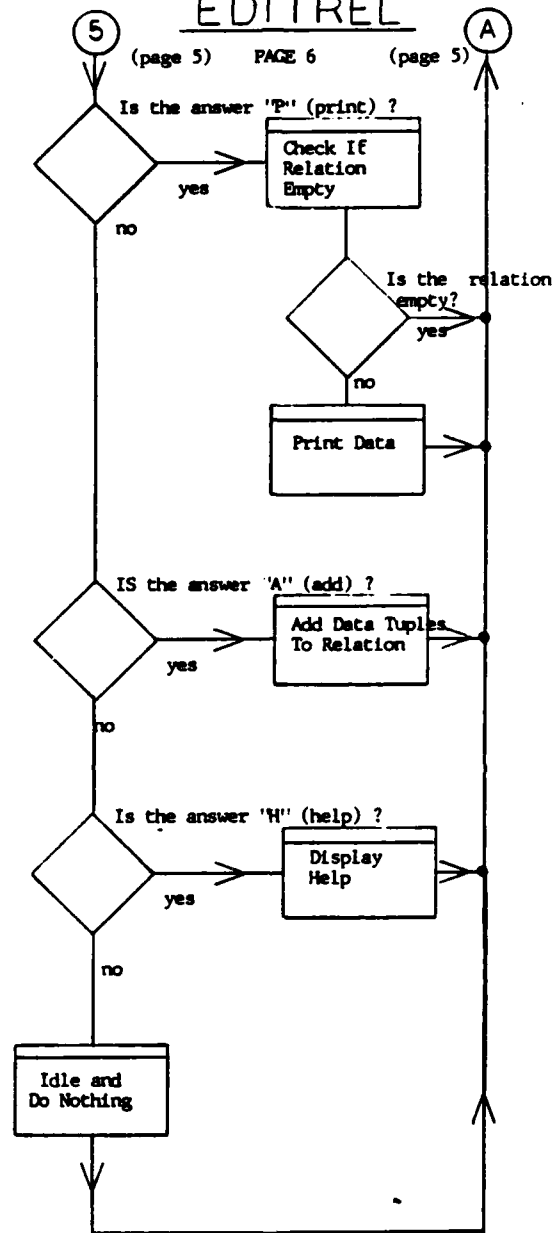


# EDITREL





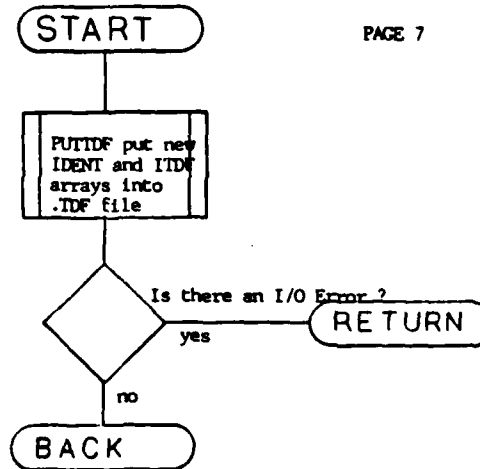
# EDITREL



## EDITREL

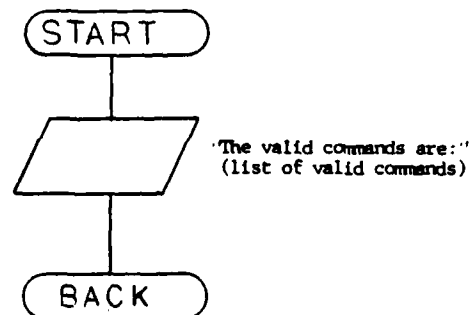
### IQUIT

PAGE 7

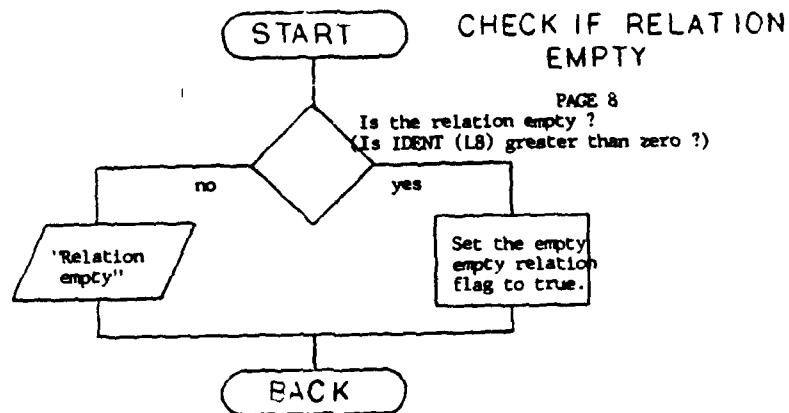


---

### DISPLAY\_HELP

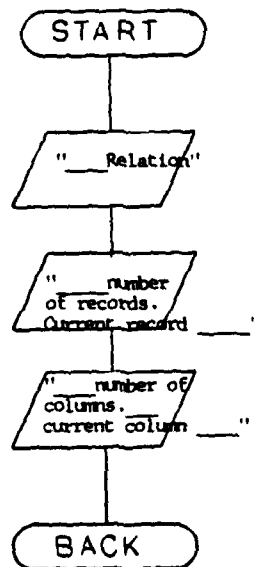


## EDITREL



---

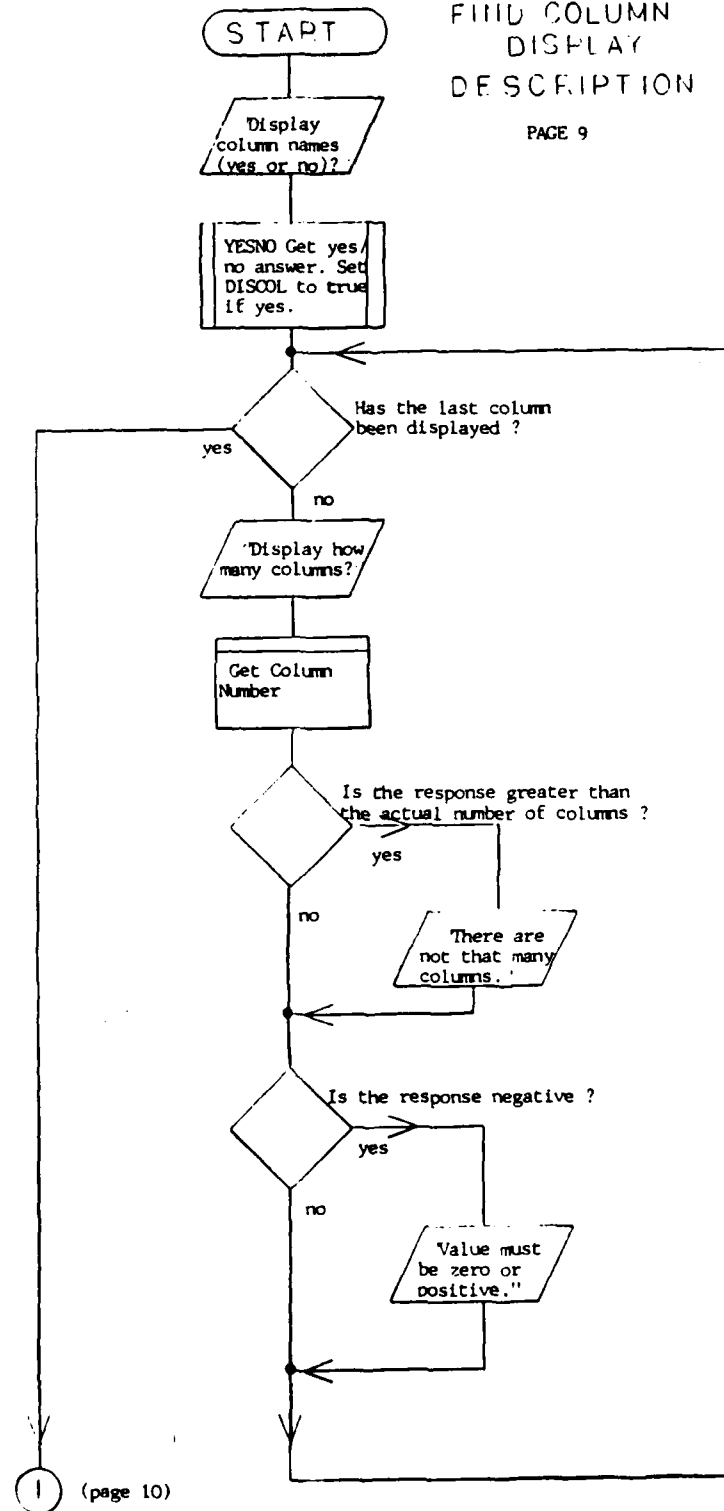
## PRESENT RELATION STATUS



EDIT F-L

FIND COLUMN  
DISPLAY  
DESCRIPTION

PAGE 9



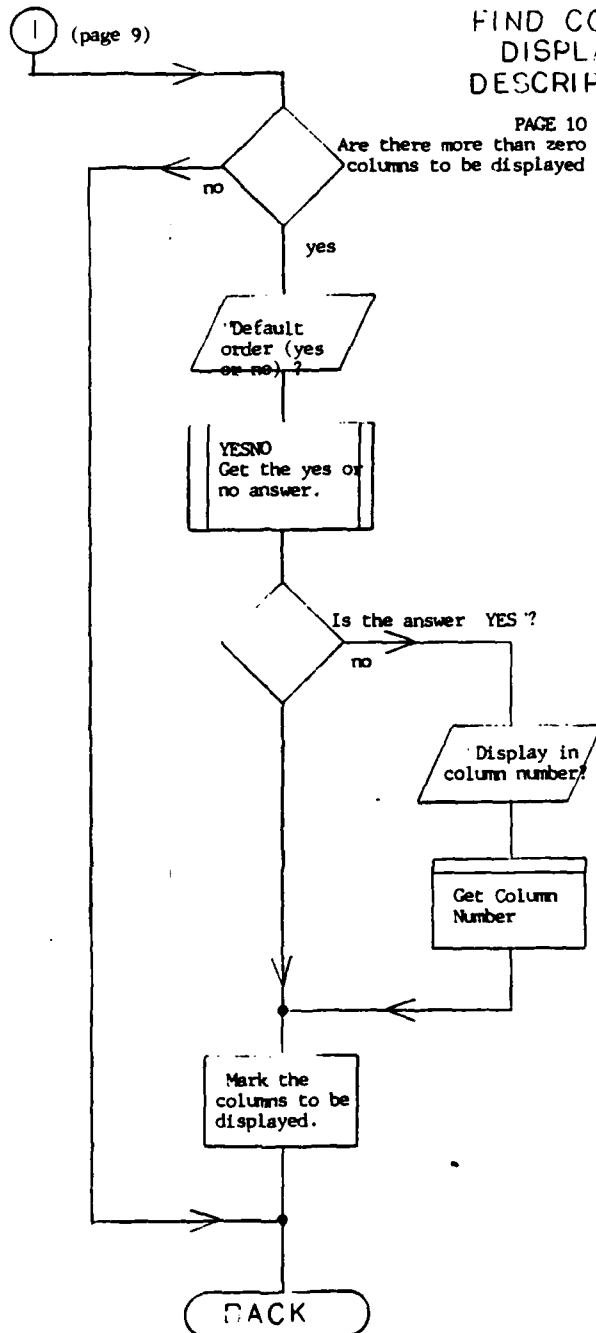
1 (page 10)

EDITREL

FIND COLUMN  
DISPLAY  
DESCRIPTION

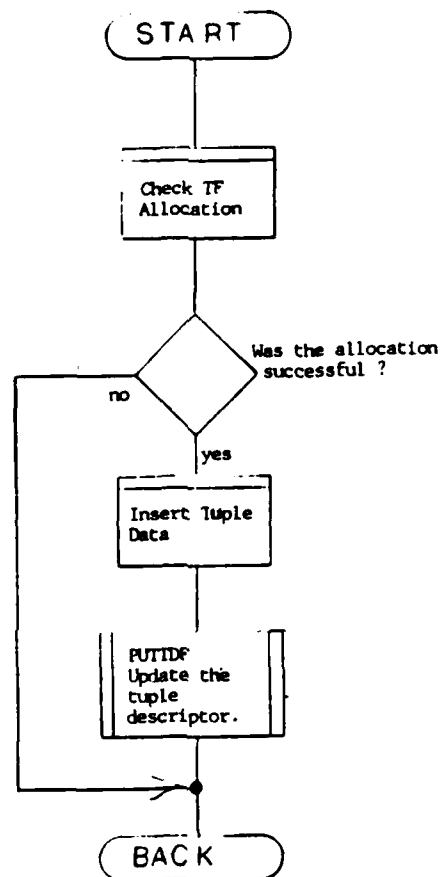
PAGE 10

Are there more than zero  
columns to be displayed ?



EDIT REL  
INSERT A TUPLE

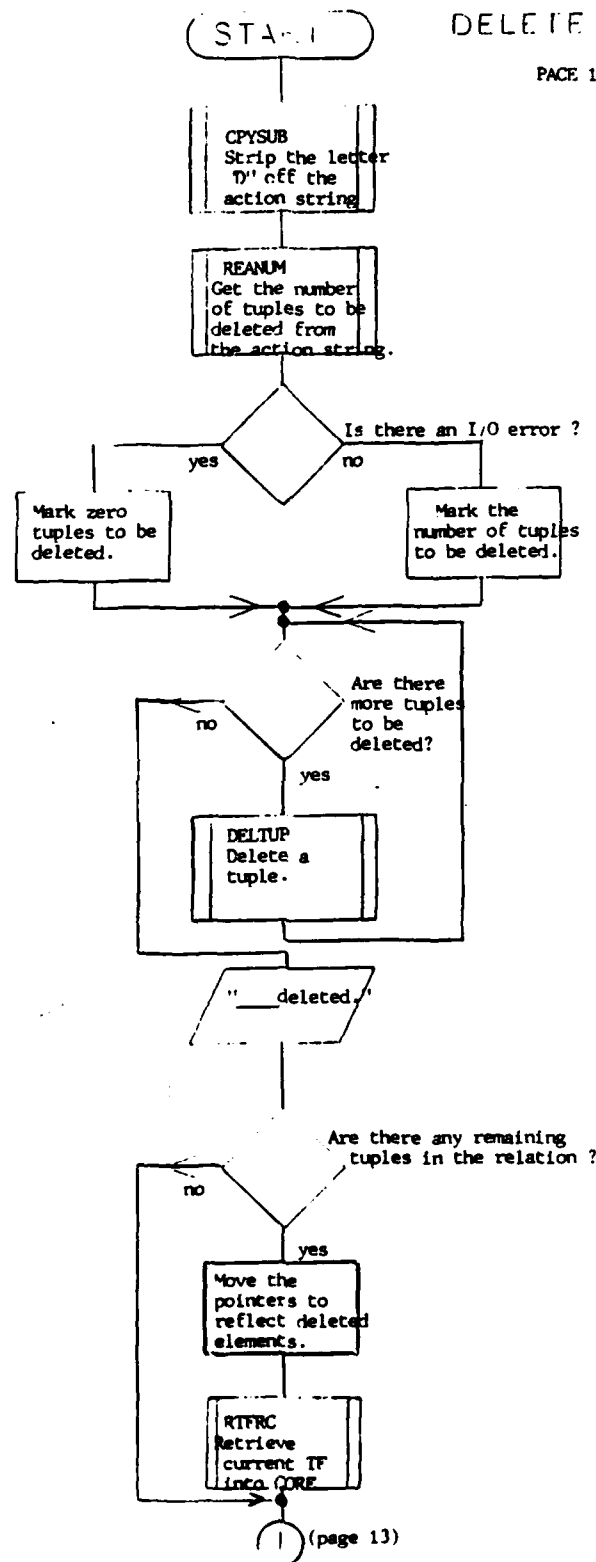
PAGE 11



# EDITREL

## DELETE TUPLES

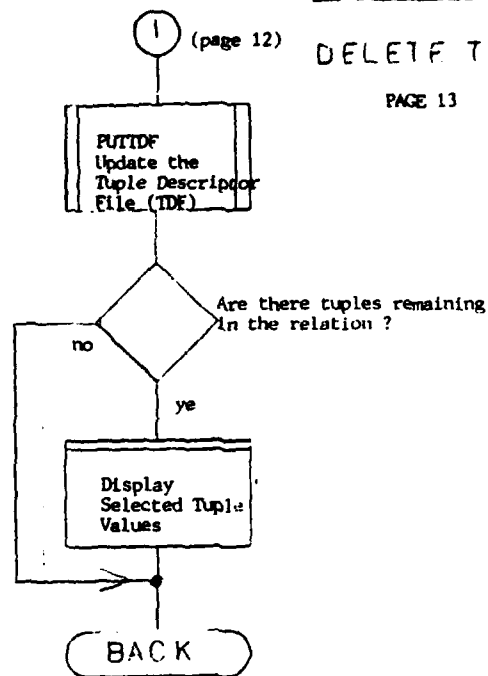
PAGE 12



## EDITREL

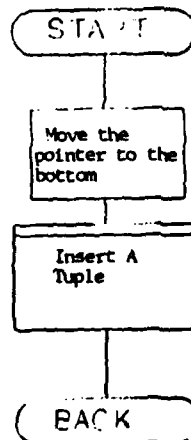
### DELETE TUPLES

PAGE 13



---

### ADD TUPLE AT BOTTOM

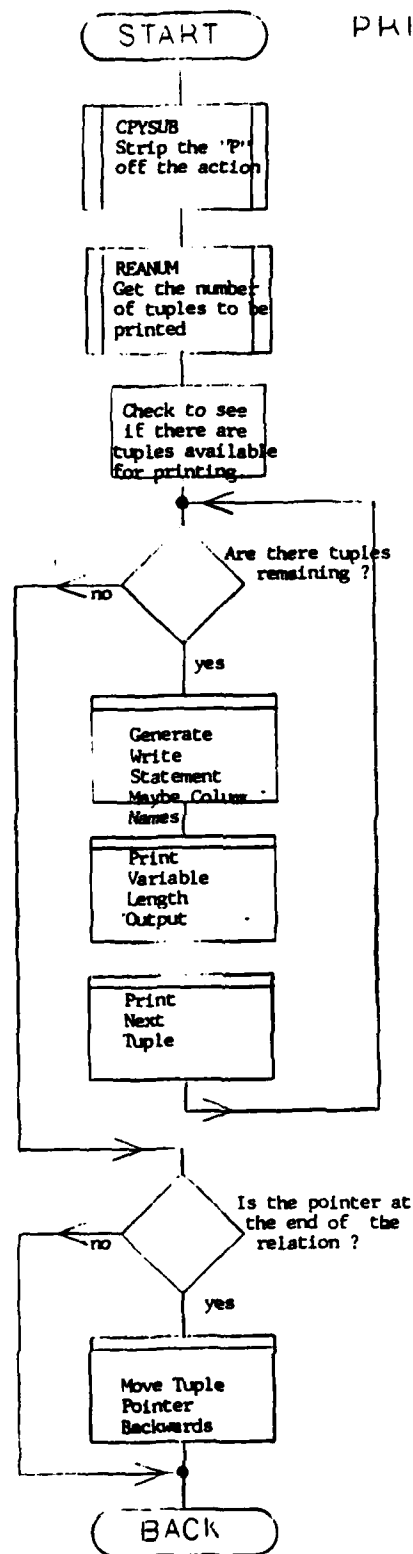




# EDITREL

PRINT DATA

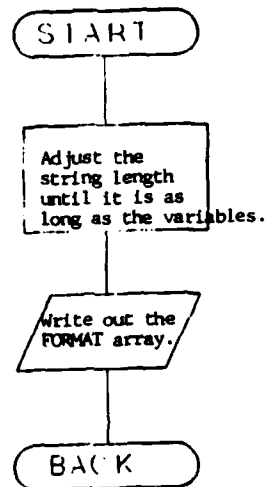
PAGE 14



EDITREL

PRINT  
VARIABLE  
LENGTH  
OUTPUT

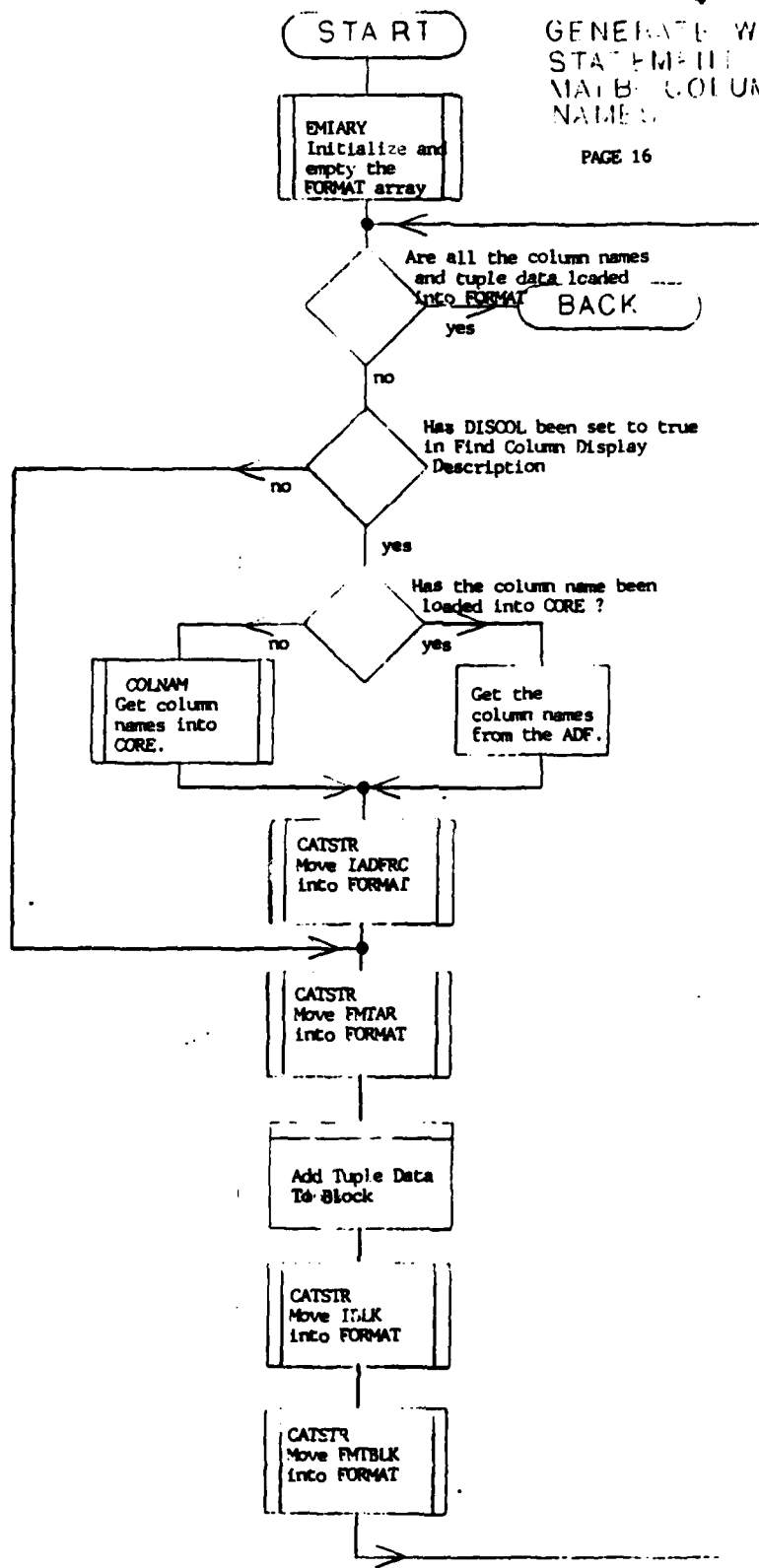
PAGE 15



EDIT REL

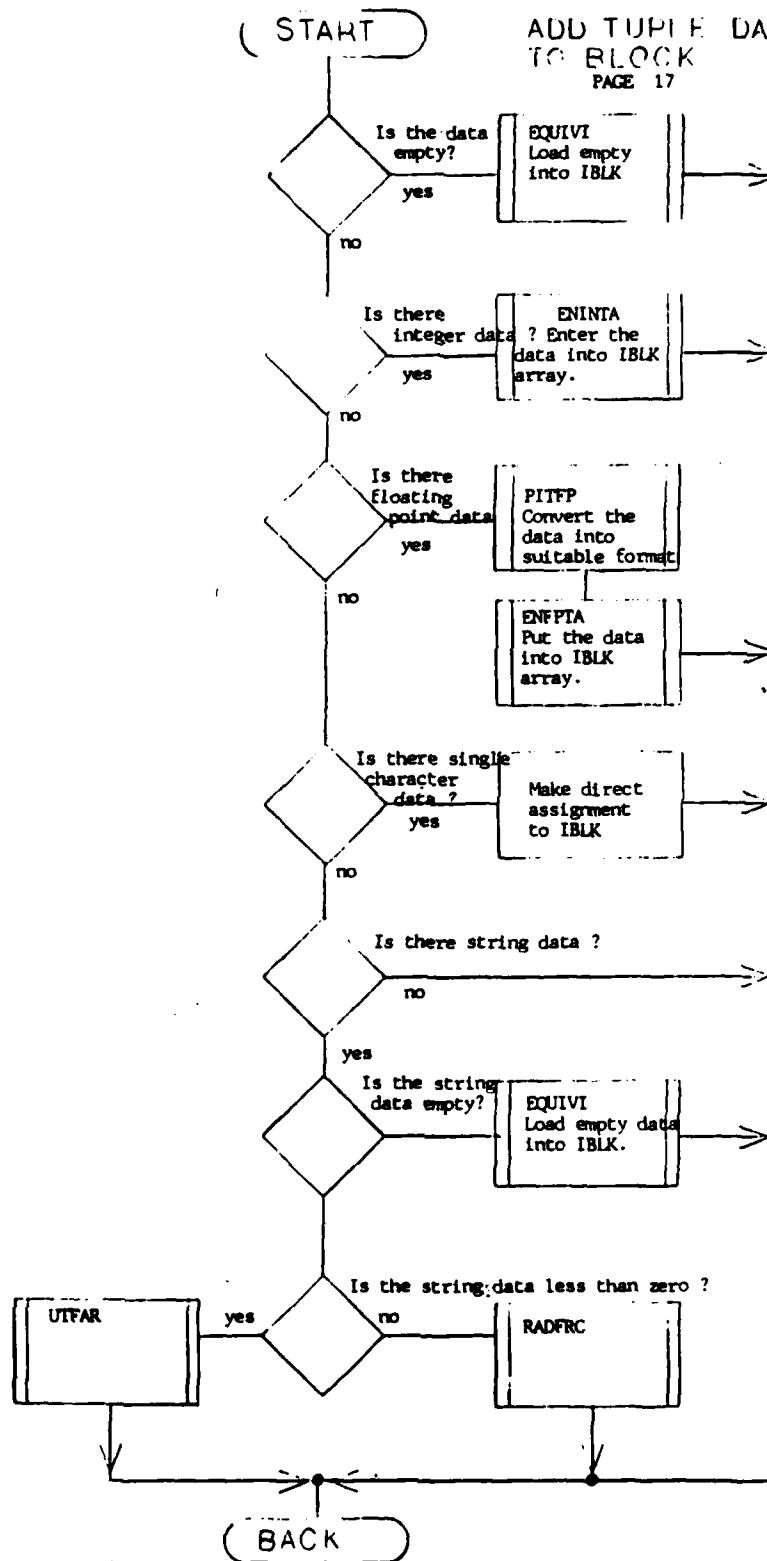
GENERATE WRITE  
STATEMENT  
MATRIX COLUMN  
NAMES

PAGE 16



# EDITREL

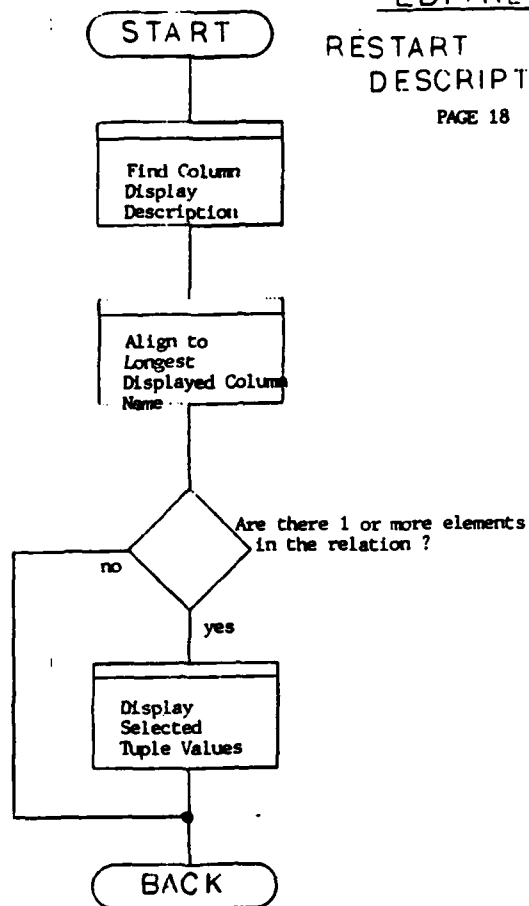
ADD TUPLE DATA  
TO BLOCK  
PAGE 17



EDITREL

RESTART  
DESCRIPTION

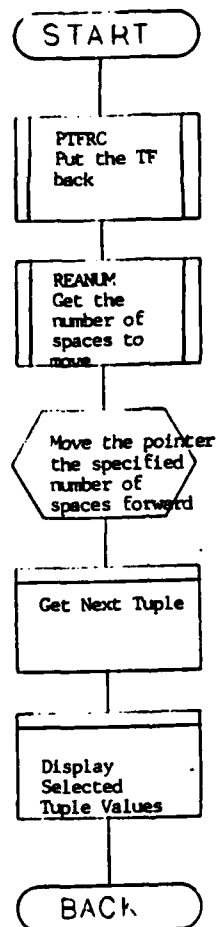
PAGE 18



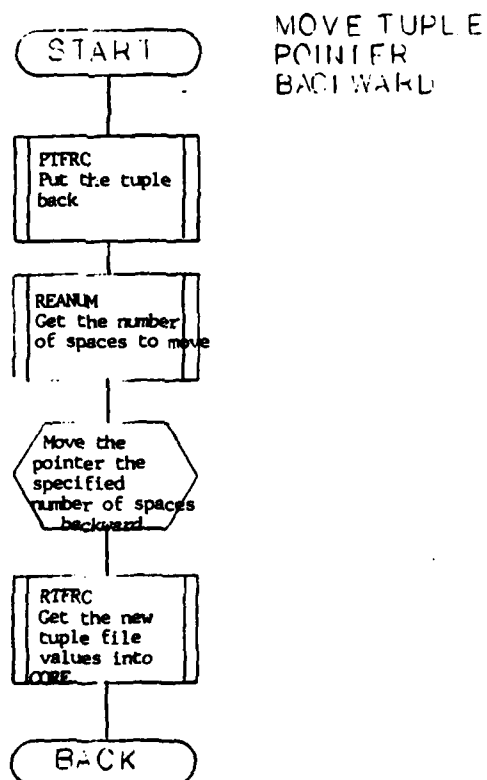
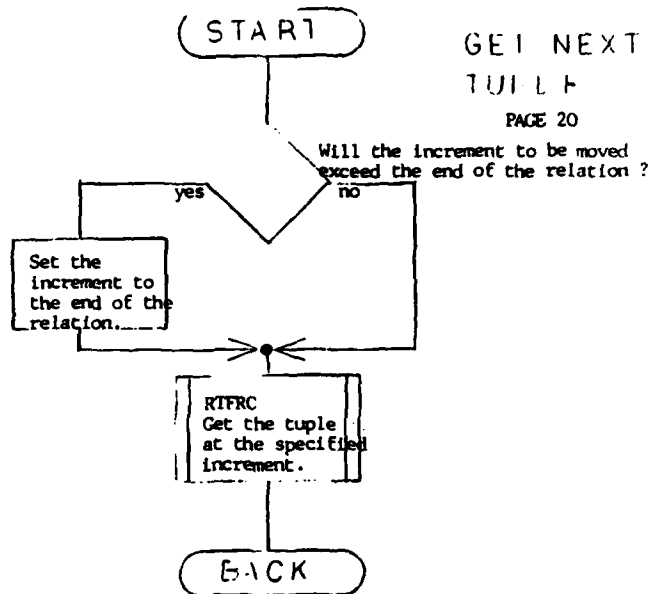
## EDIT REL

MOVE TUPLE  
POINTER  
FORWARD

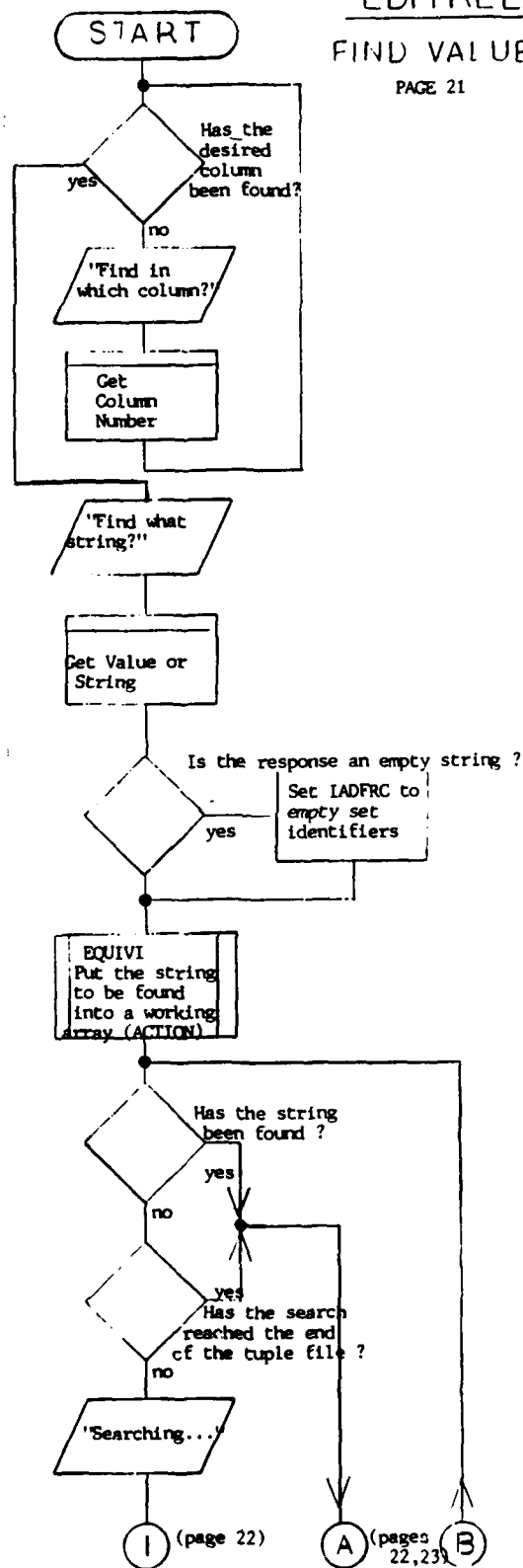
PAGE 19



EDITREL



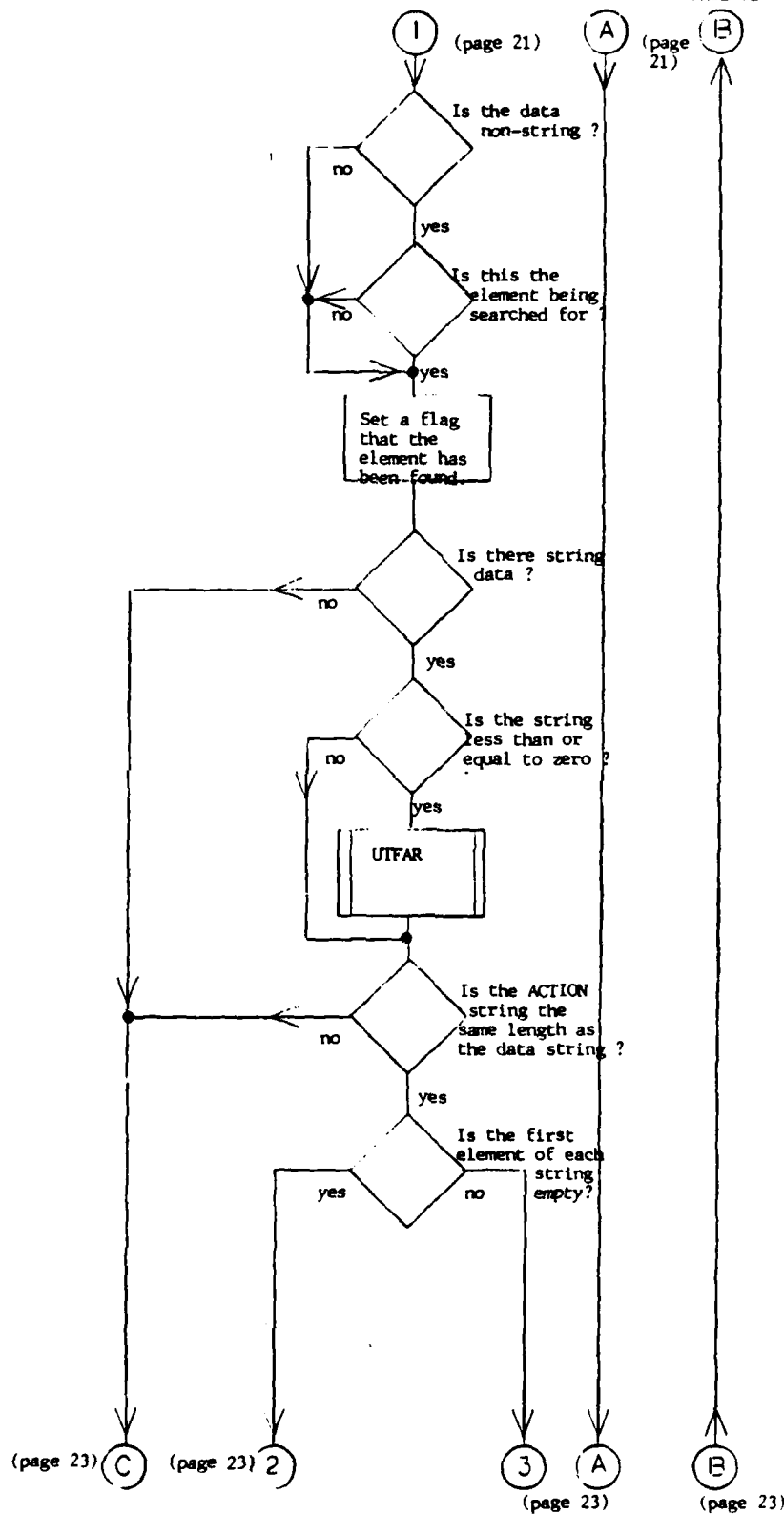
EDITRLL  
FIND VALUE  
PAGE 21

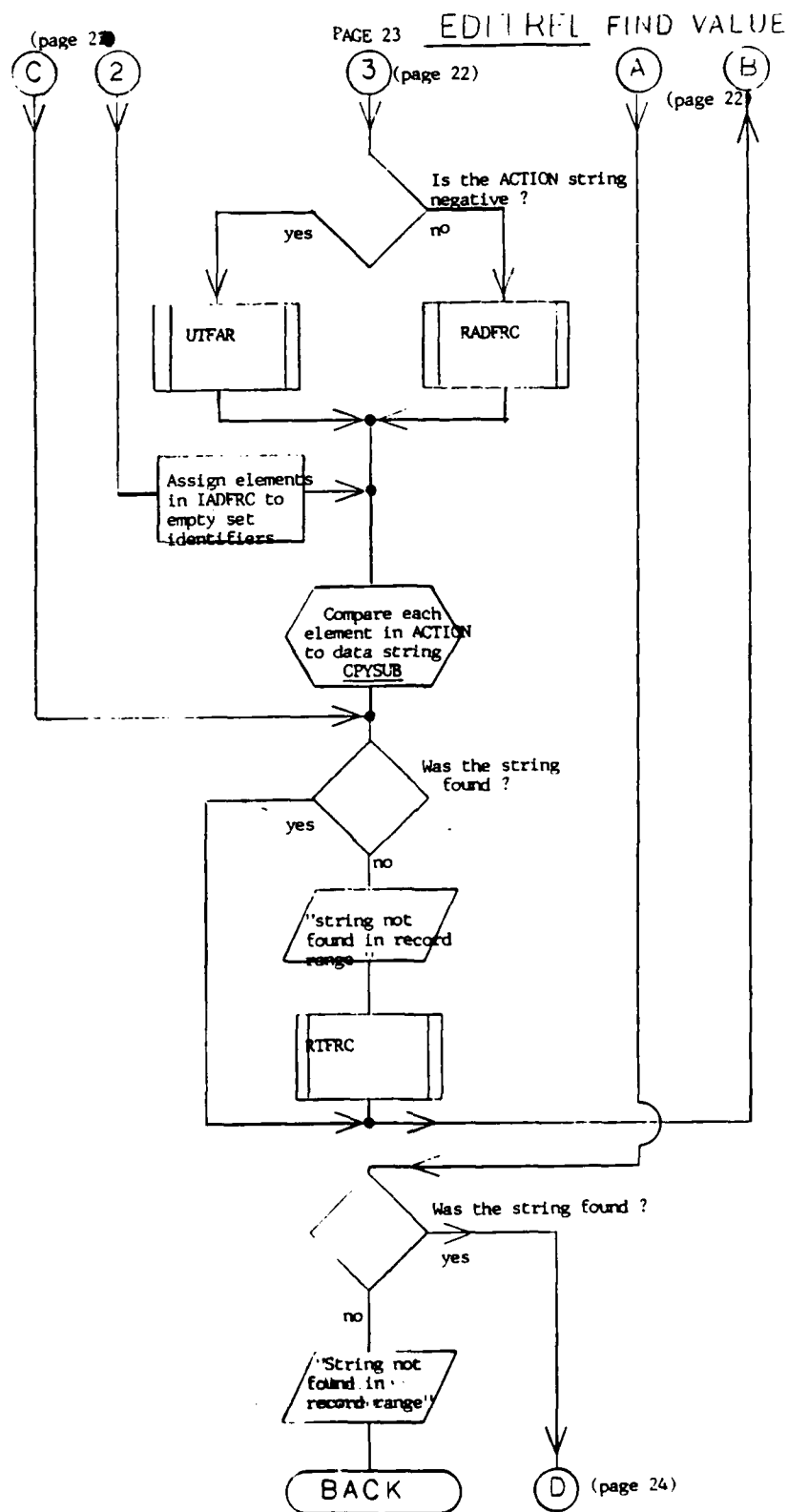




# EDITREL FIND VALUE

PAGE 22



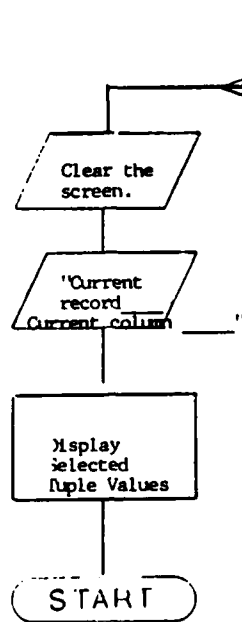


EDIT REL

FIND VALUE

(D (page 23)

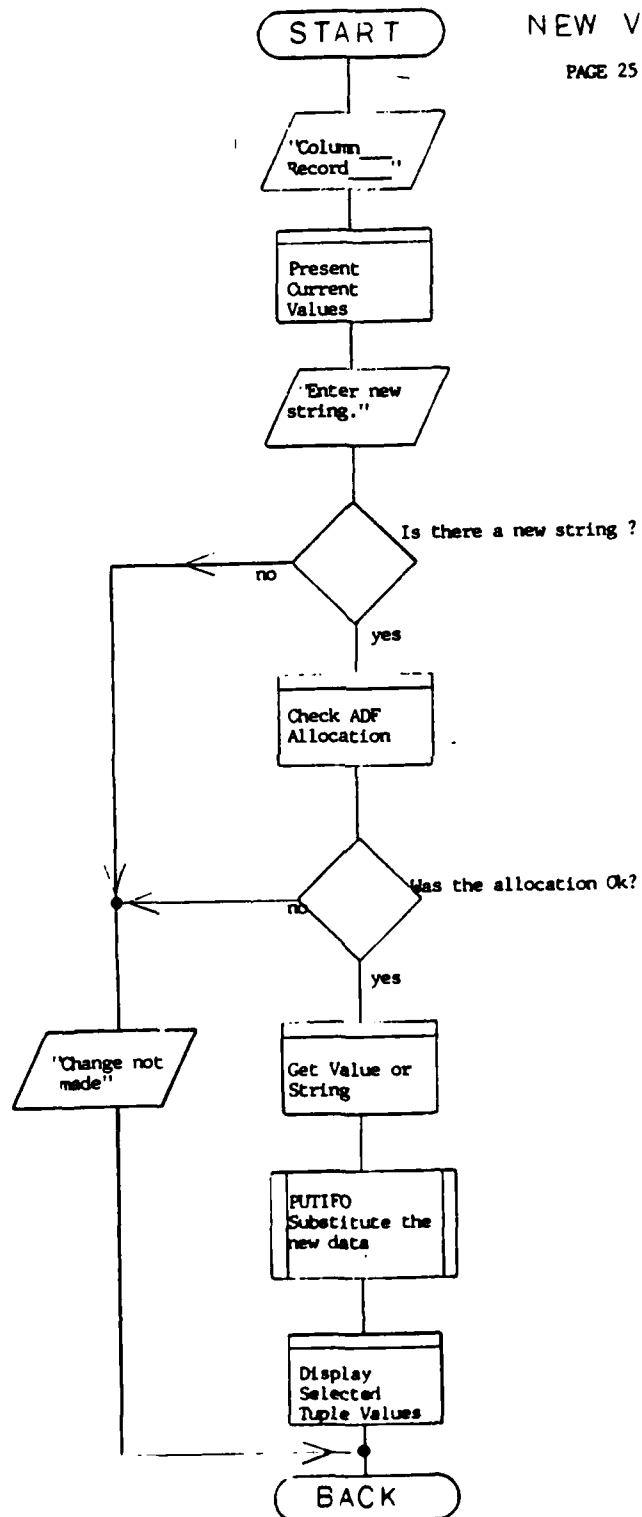
PAGE 24



# EDITREL

## SUBSTITUTE NEW VALUE

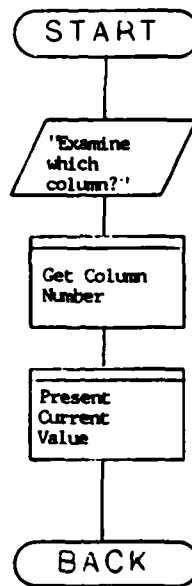
PAGE 25



EDITREL

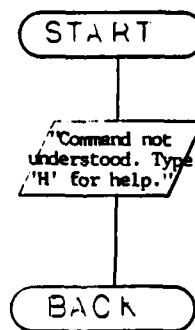
EXAMINE  
CURRENT  
VALUES

PAGE 26



---

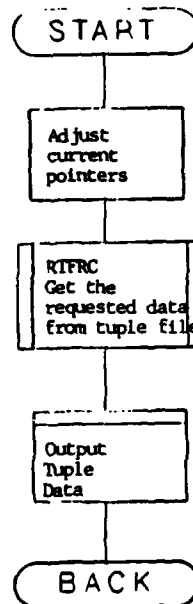
IDLE AND DO NOTHING



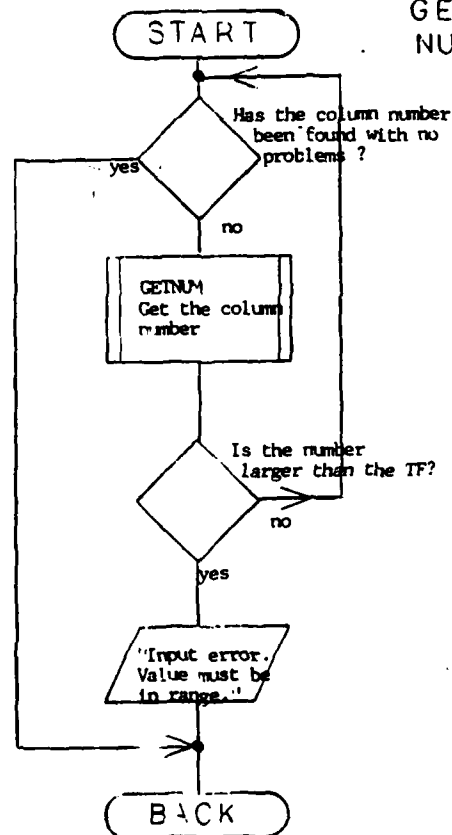
# EDITREL

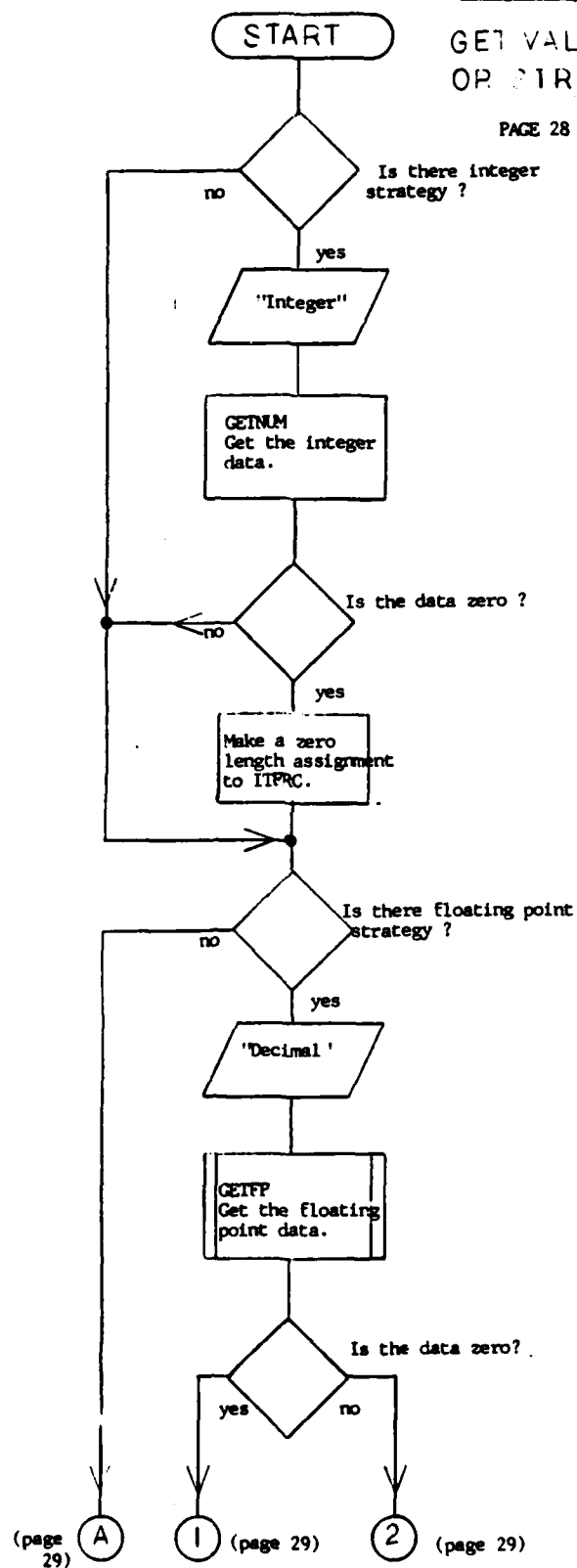
PRESENT  
CURRENT  
VALUE

PAGE 27



## GET COLUMN NUMBER

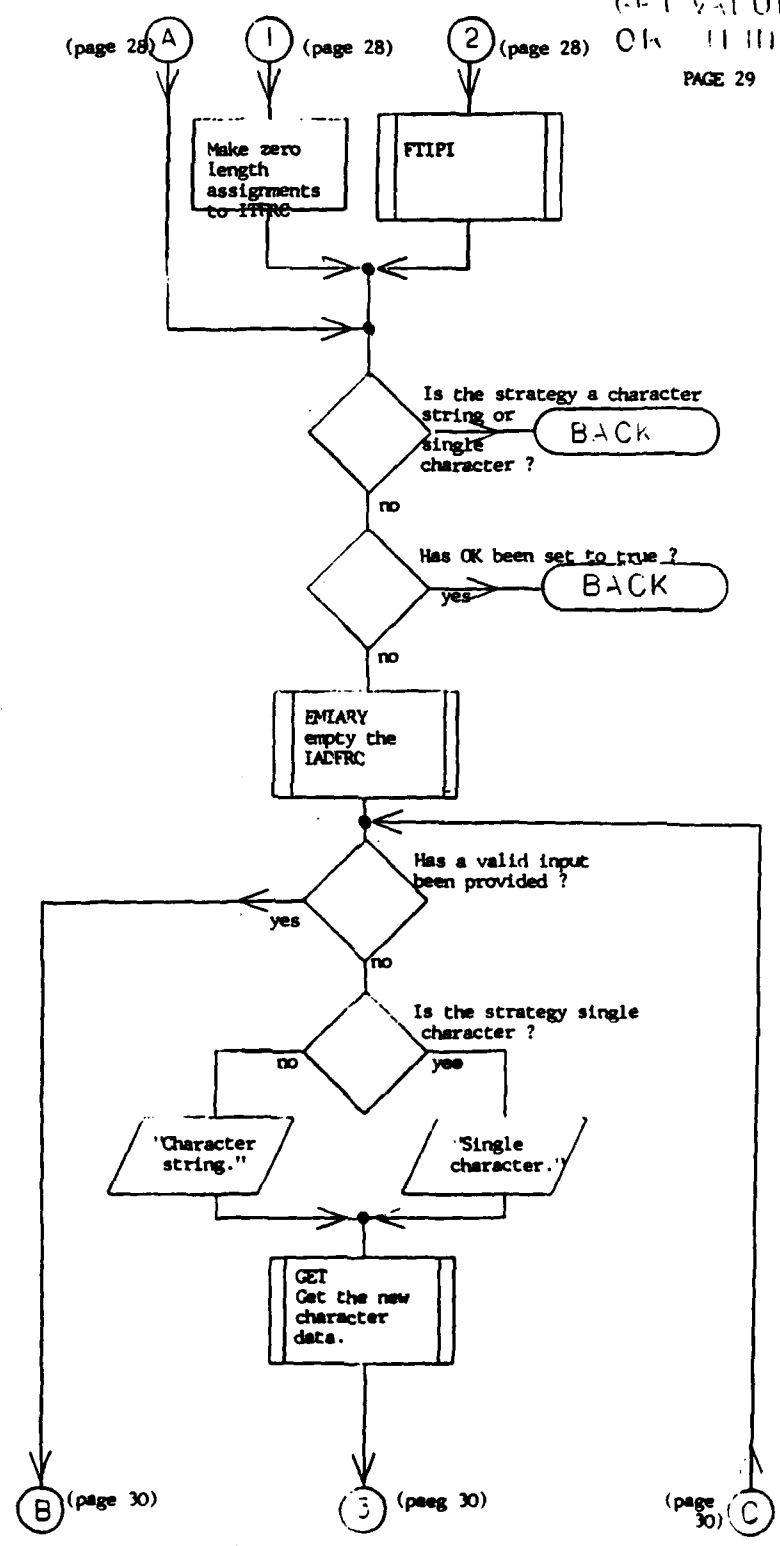




EDITED

GET VALUE  
OR  
TIME

PAGE 29





# EDITREL

GET VALUE  
OR STRING

PAGE 30

(B) (page 29)

(page 29) 3

(C) (page 29)

STRLEN  
determine  
string  
length

Is the string of zero length ?

yes

Set the  
value to zero.

no

Adjust length  
qualifiers  
and make  
assignments

CPYSUB  
Copy the adjusted  
data in.

CATSTR

Is there single  
character strategy ?

no

yes

Is the data of length one ?

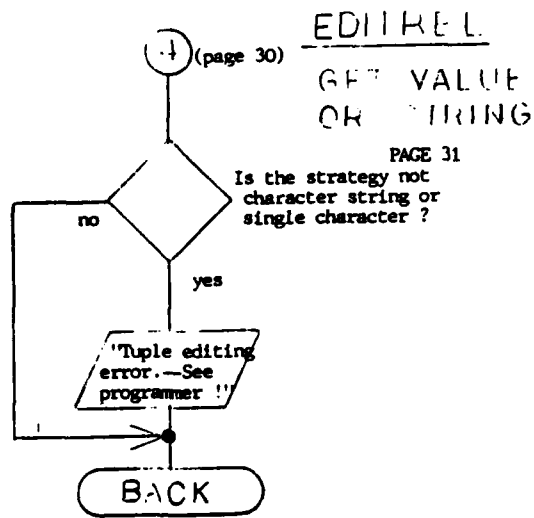
yes

no

CPYSUB

"Single  
character  
only ."

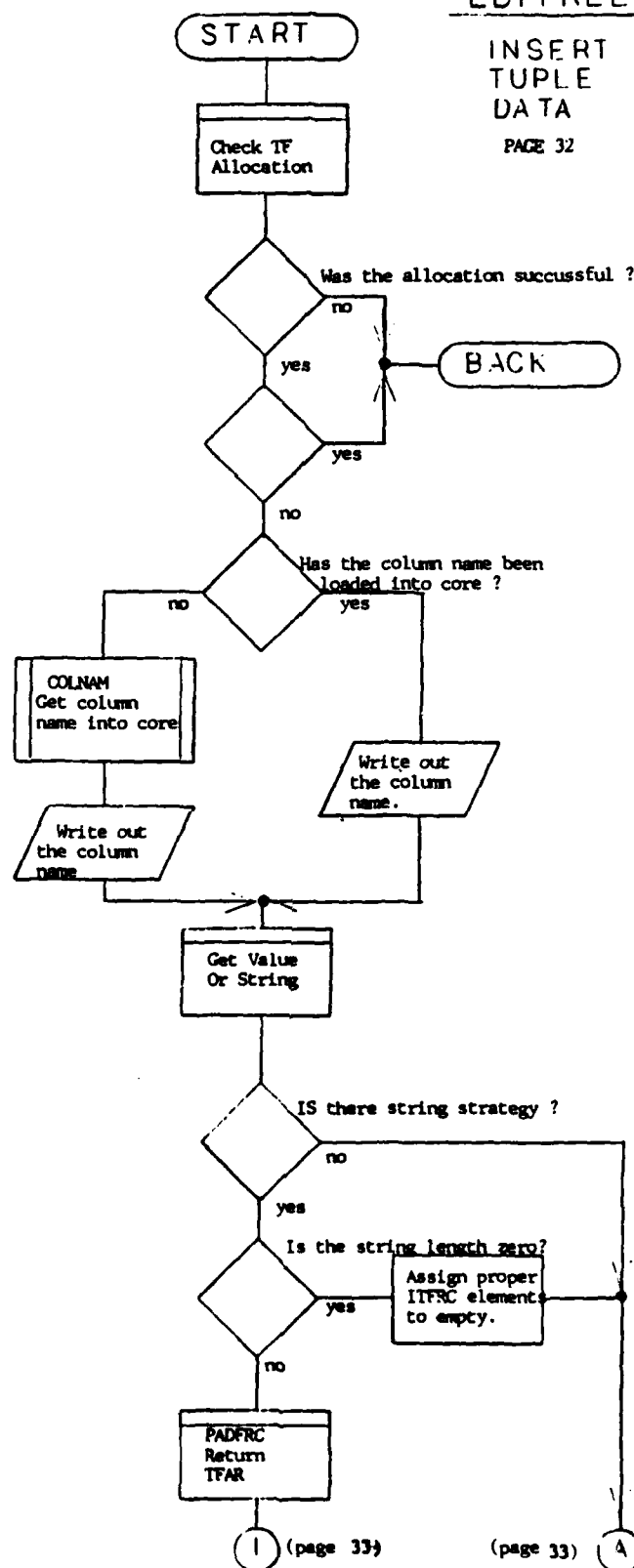
4 (page 31)



# EDITREL

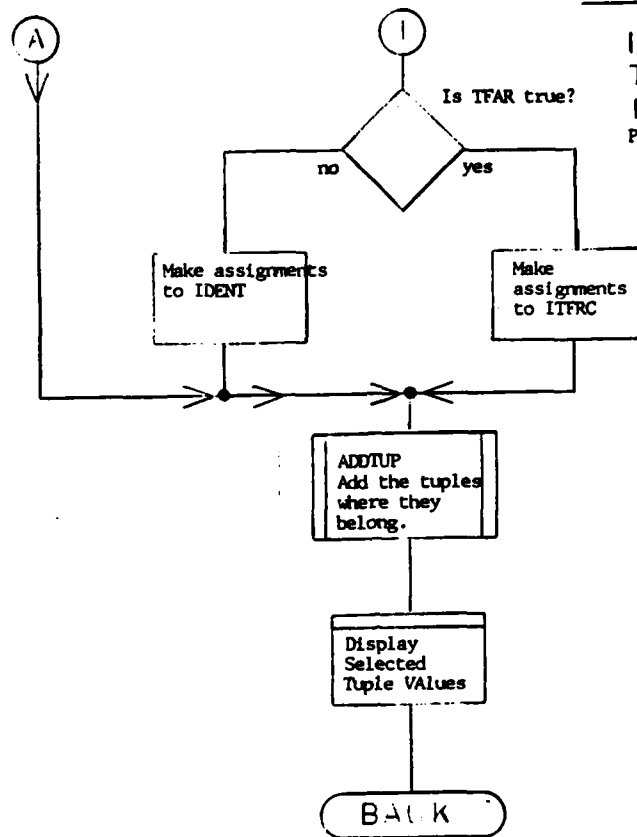
INSERT  
TUPLE  
DATA

PAGE 32



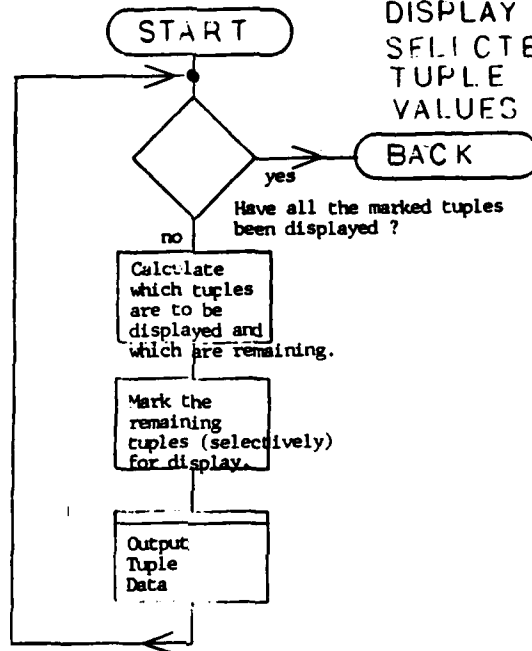
EDITREL

INSERT  
TUPLE  
DATE  
PAGE 33

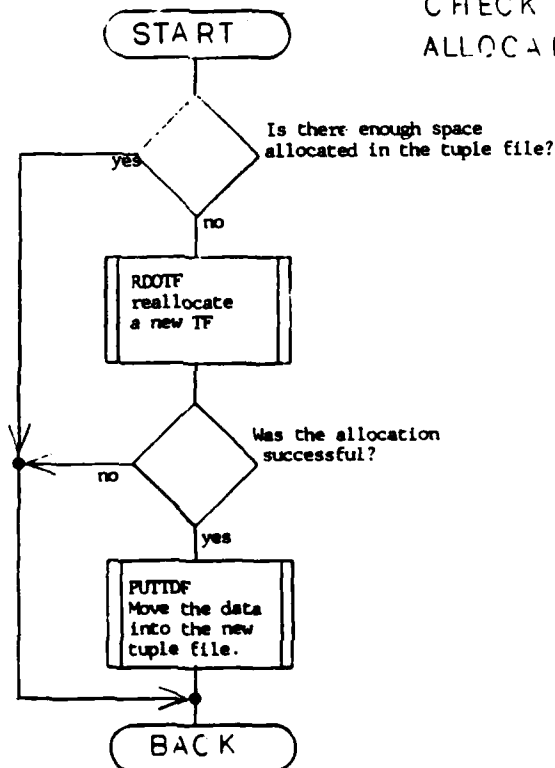


## EDITREL

DISPLAY  
SELECTED  
TUPLE  
VALUES PAGE 34



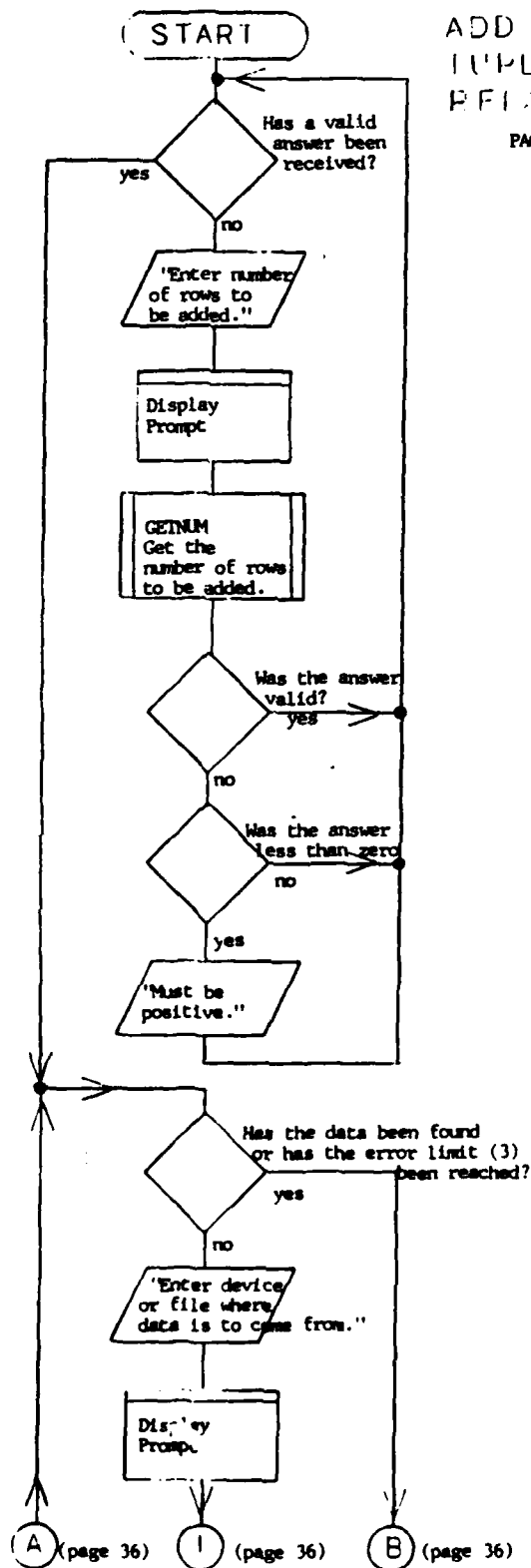
## CHECK TF ALLOCATION



EDITED

# ADD DATA TUPLES TO RELATION

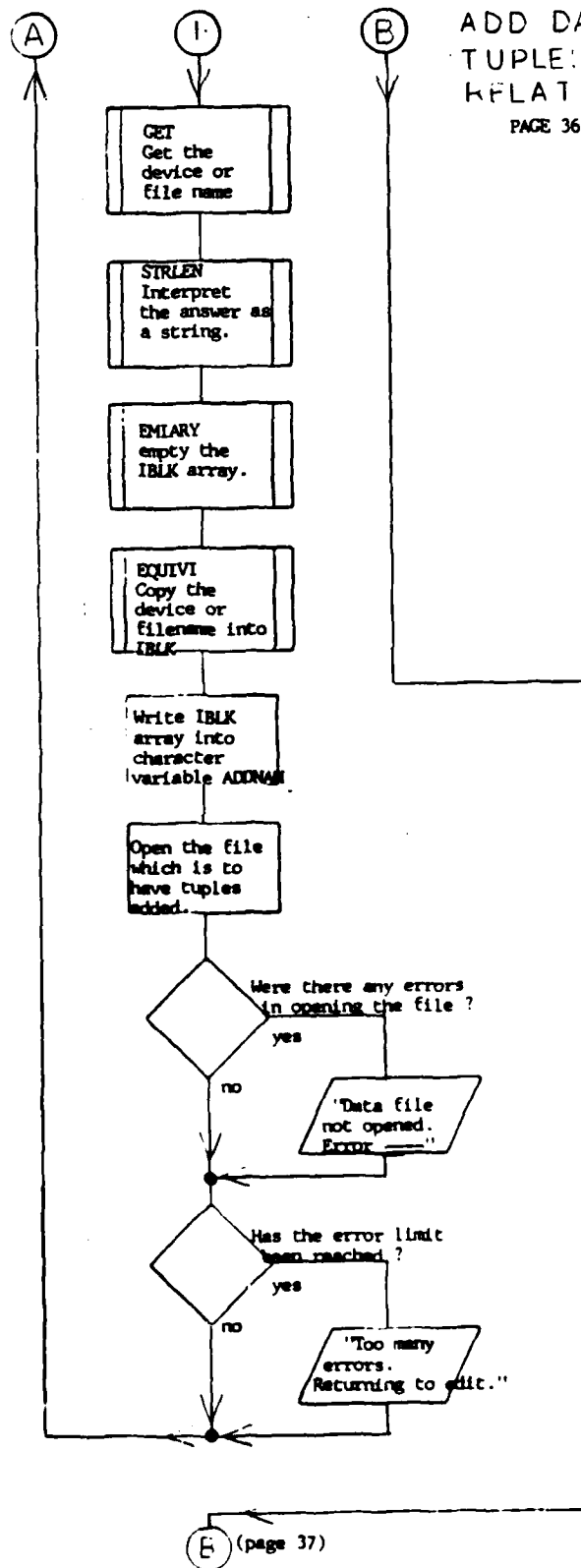
PAGE 35



# EDITREL

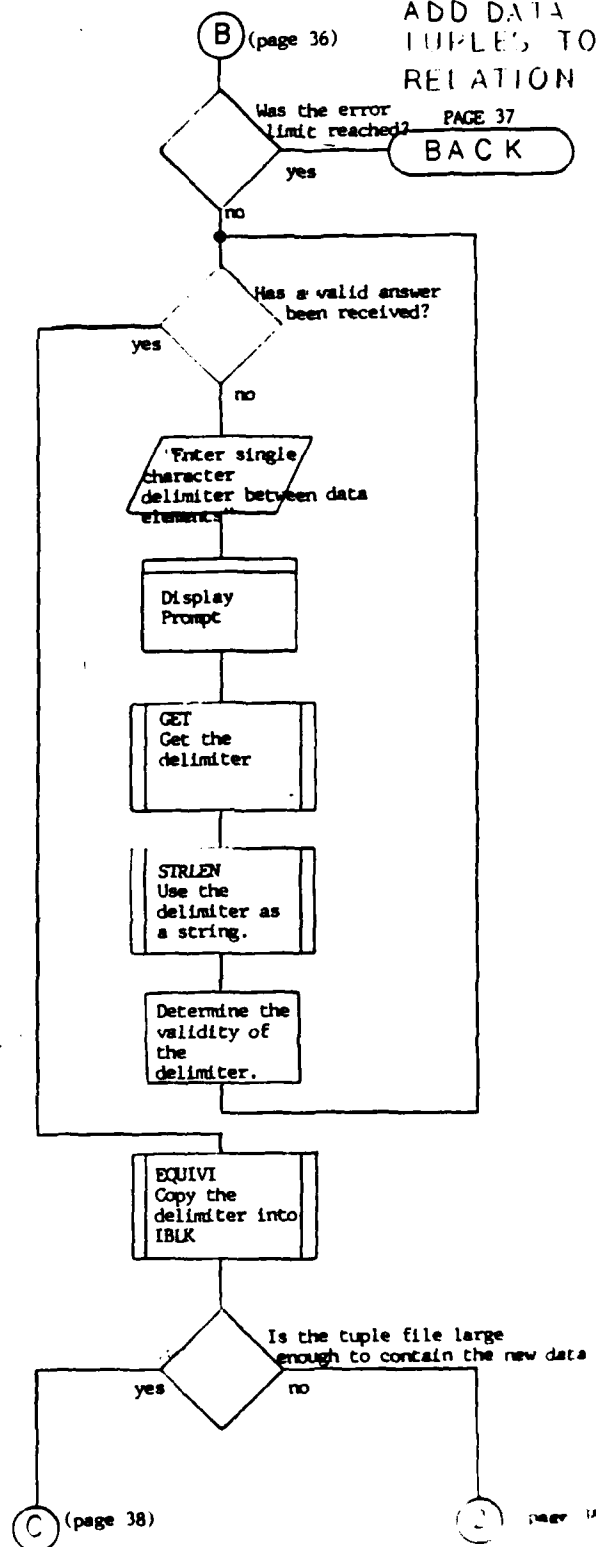
## ADD DATA TUPLES TO RELATION

PAGE 36



# EDITREL

## ADD DATA TUPLES TO RELATION





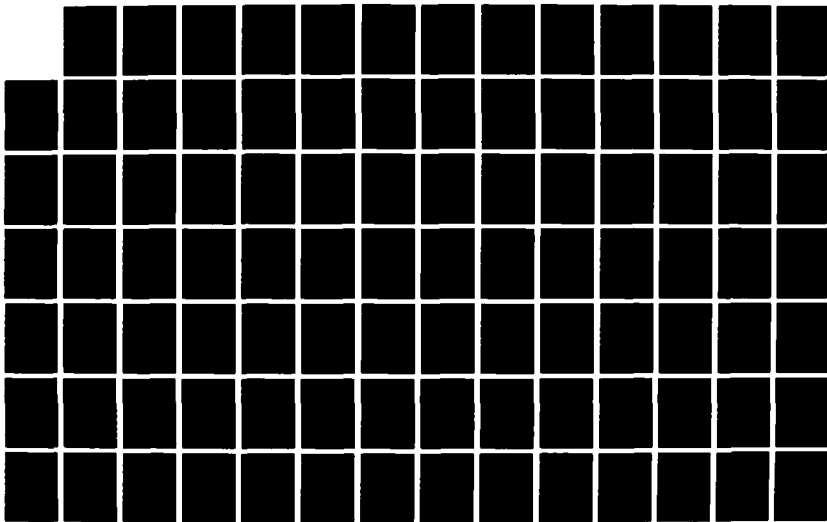
AD-A188 002

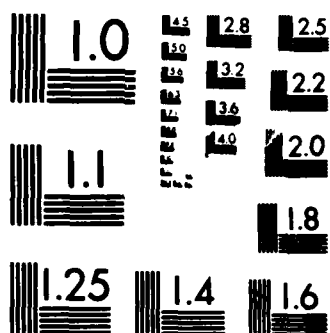
FEASIL IMPLEMENTATION UNDER VAX VMS WITH DESIGN  
INFORMATION(U) ALABAMA UNIV IN HUNTSVILLE DEPT OF  
ELECTRICAL AND COMPUTER EN. J D HARR ET AL. NOV 86  
UAH-5-31325 ANSHI-CR-RD-55-86-5 F/G 12/5

3/4

UNCLASSIFIED

NN



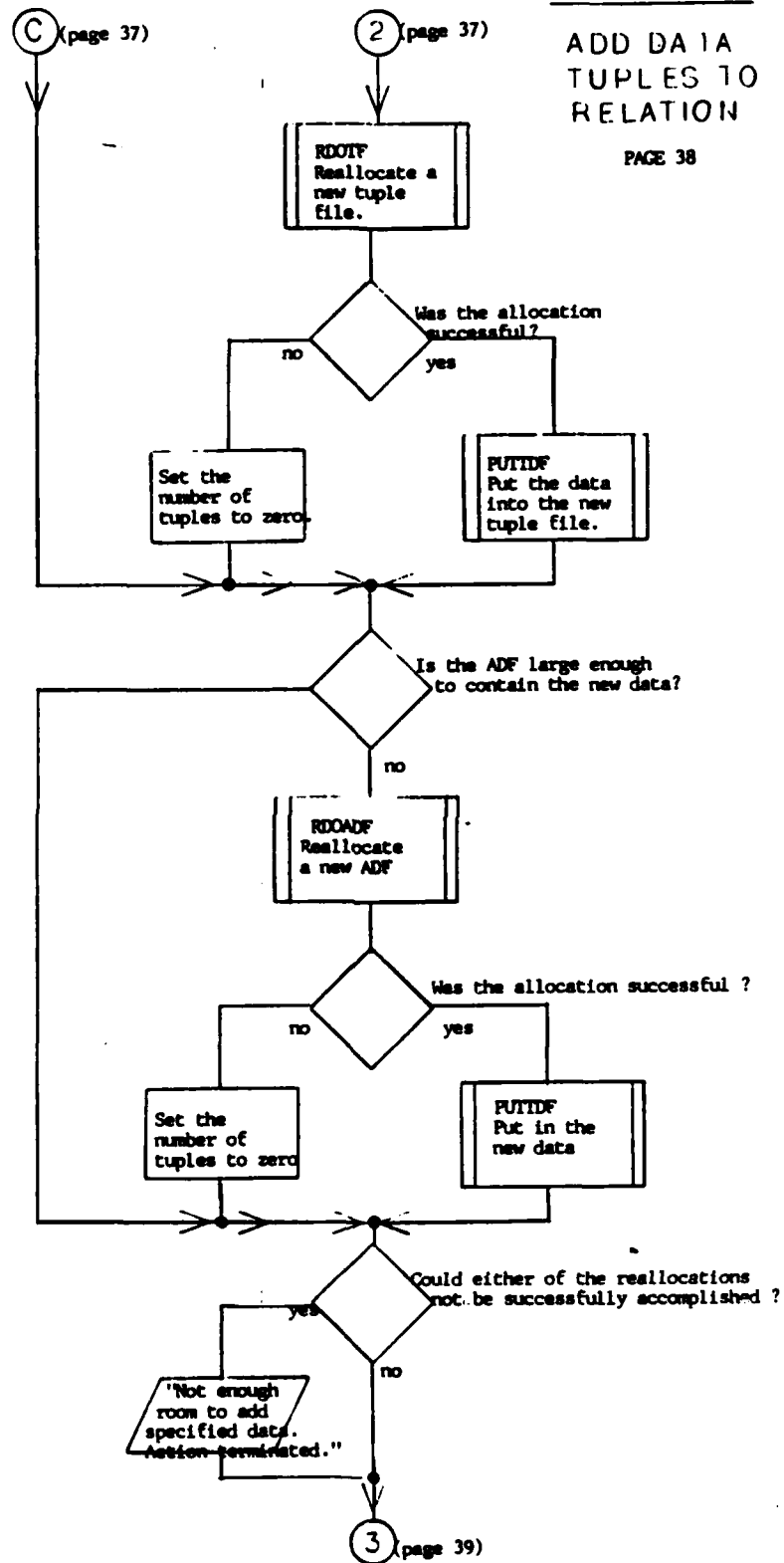


MICROCOPY RESOLUTION TEST CHART  
NATIONAL BUREAU OF STANDARDS-1963-A

# EDITREL

## ADD DATA TUPLES TO RELATION

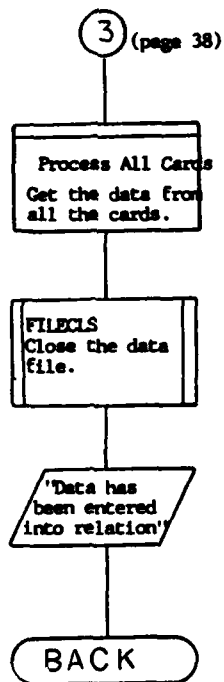
PAGE 38



EDITREL

ADD DATA  
TUPLES TO  
RELATION

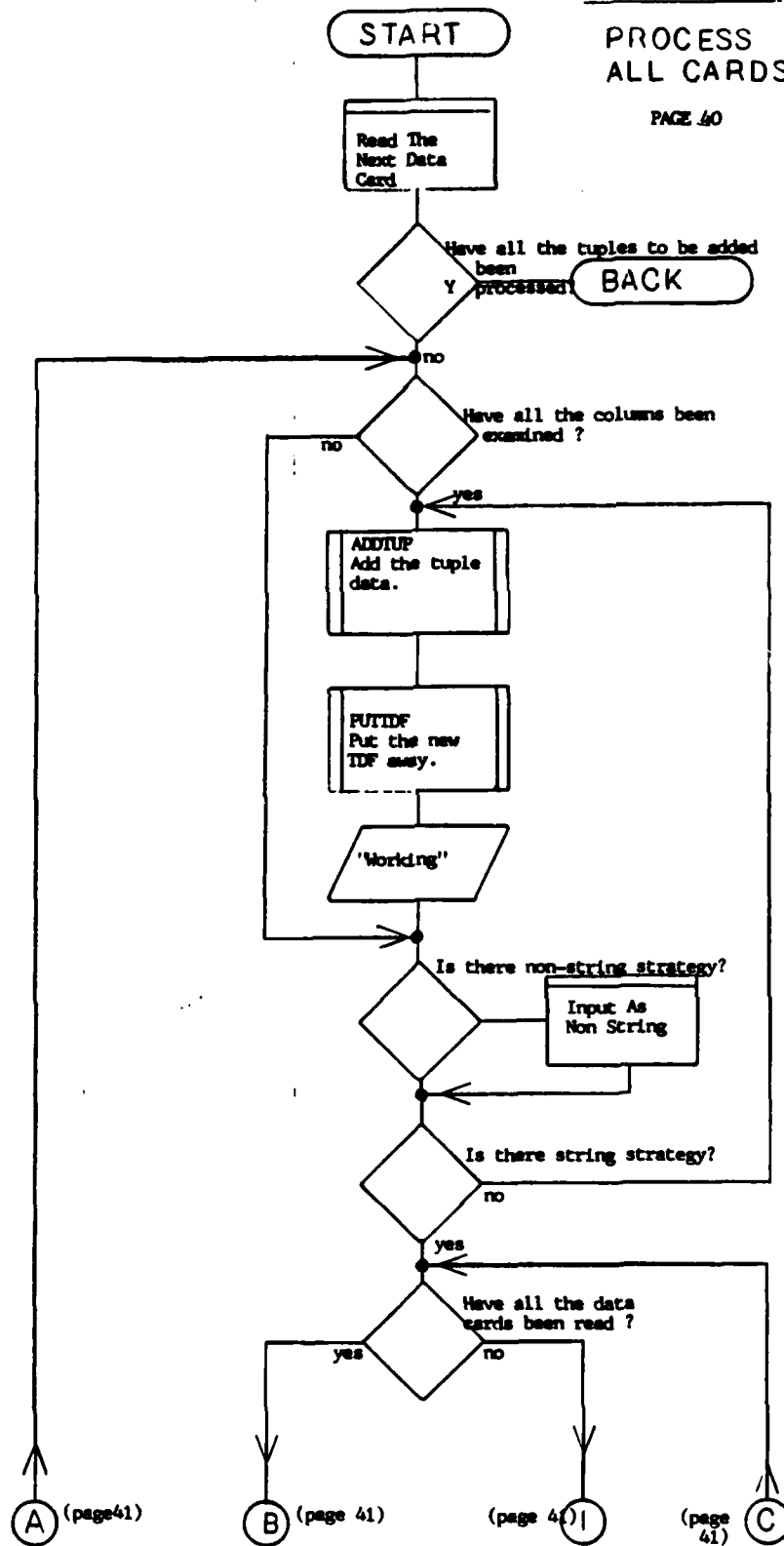
PAGE 39



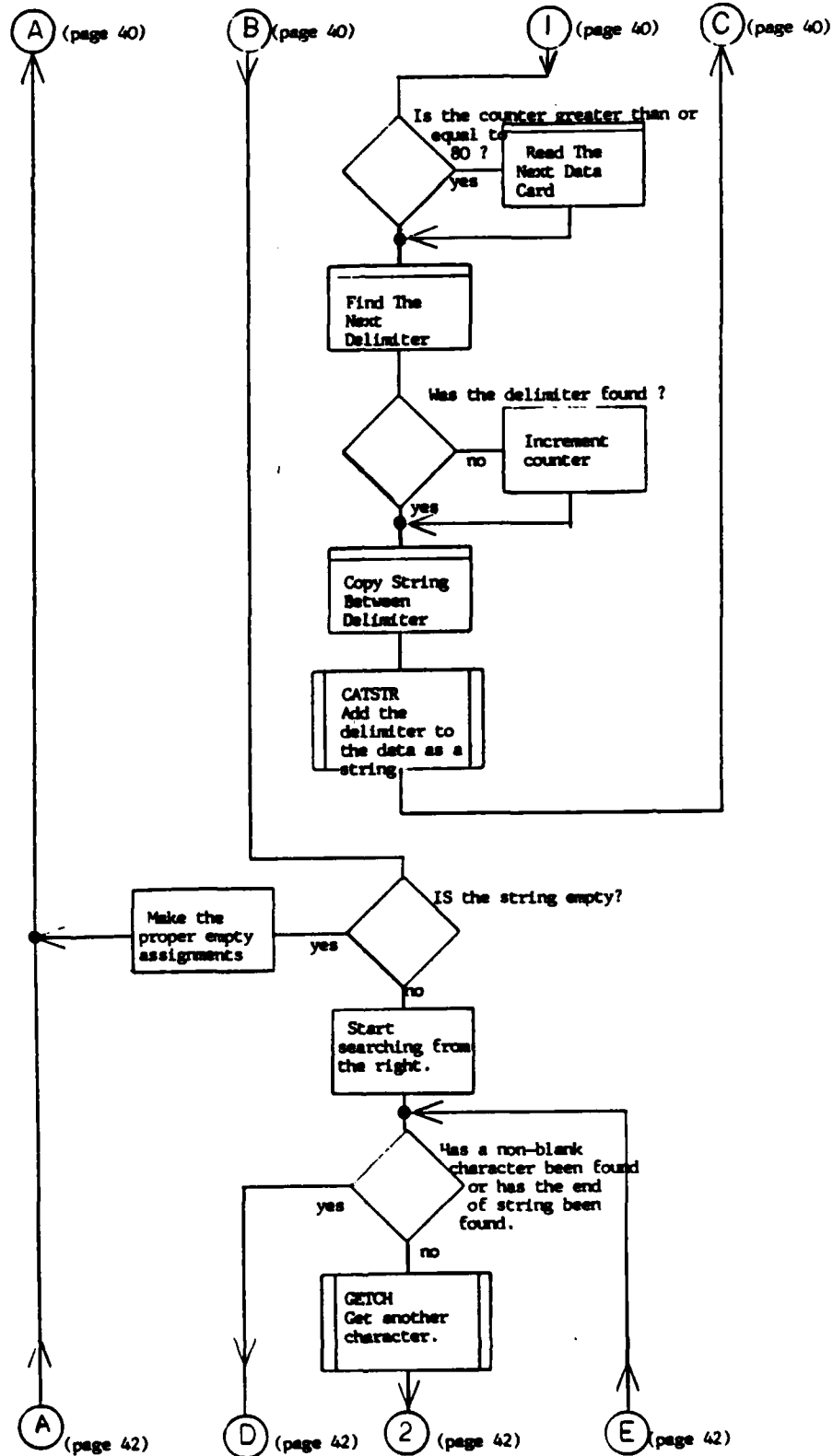
# EDITHEL

PROCESS  
ALL CARDS

PAGE 40



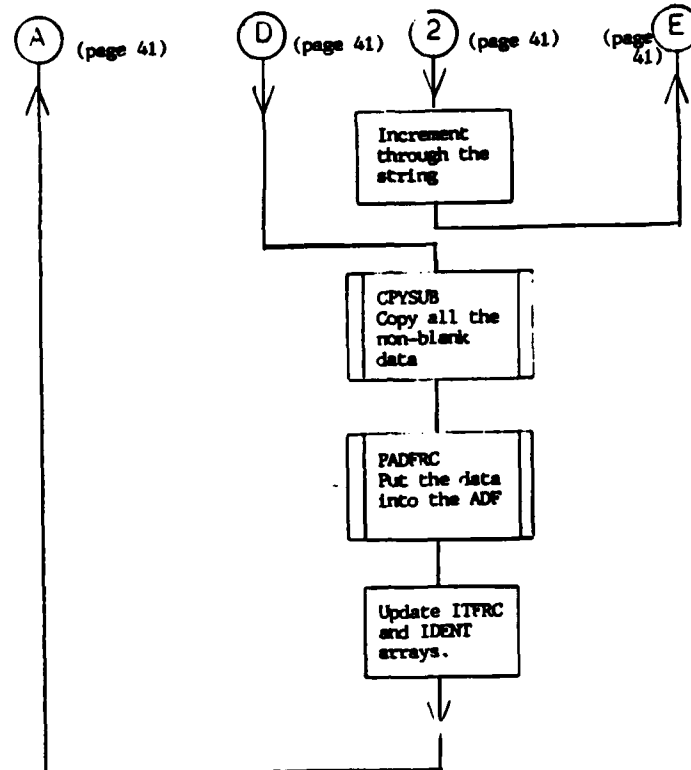
EDITALL PAGE 41  
PROCESS ALL CARDS

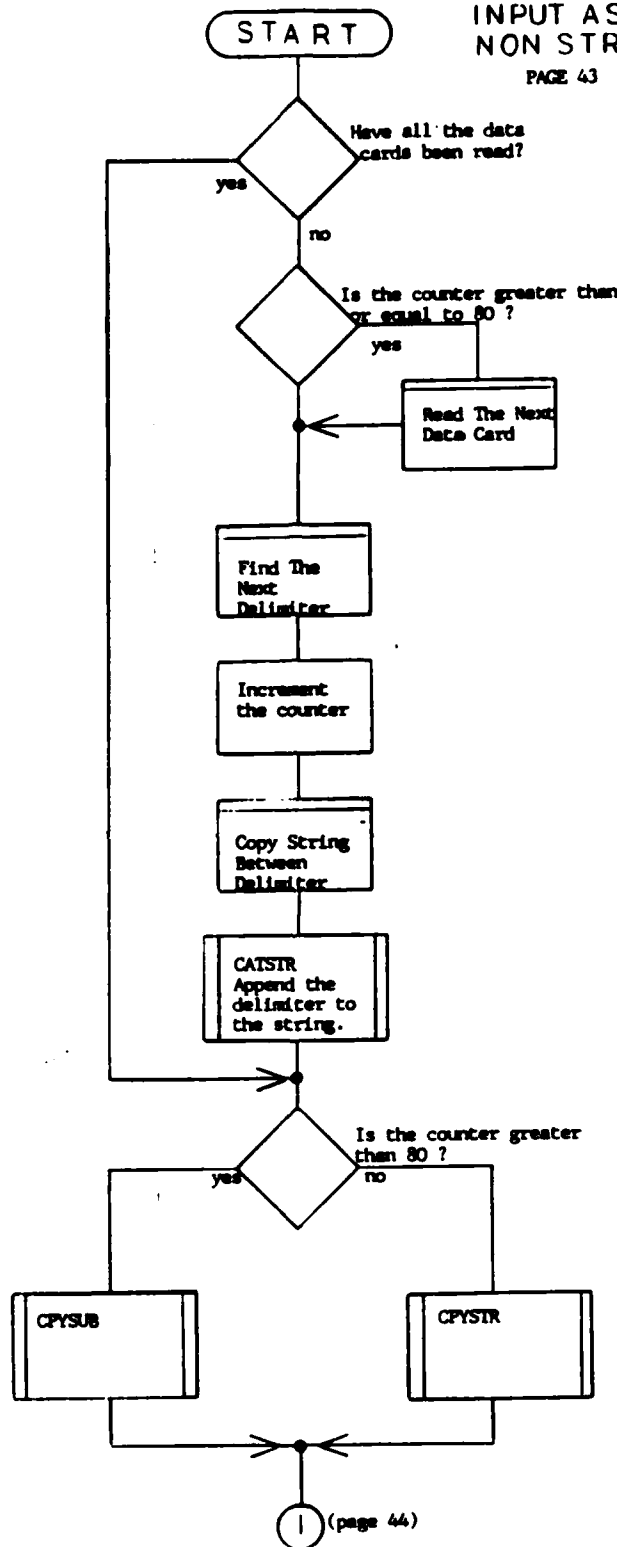


EDIT REL

PROCESS  
ALL  
CARDS

PAGE 42

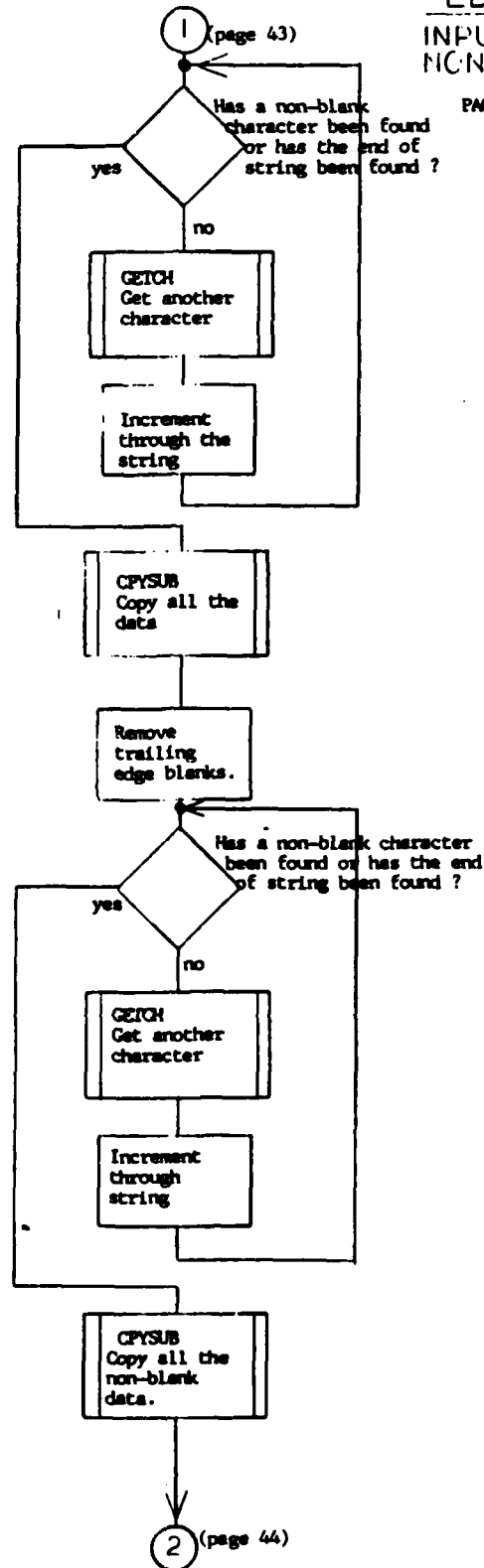






EDITREL  
INPUT A'S  
NON-STRING

PAGE 44

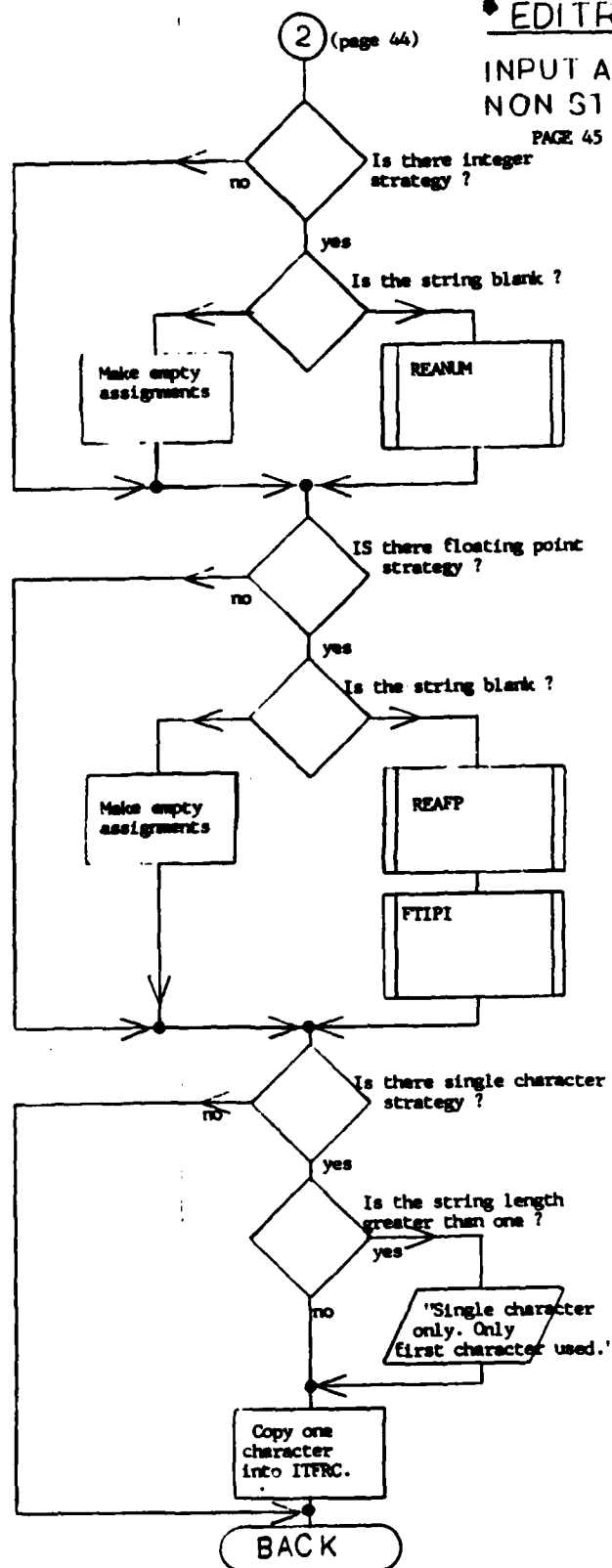


2 (page 44)

# EDITREL

INPUT AS  
NON STRING

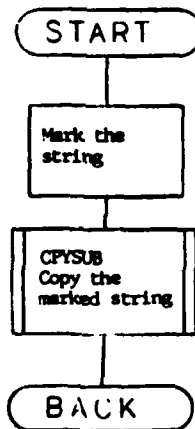
PAGE 45



EDITREL

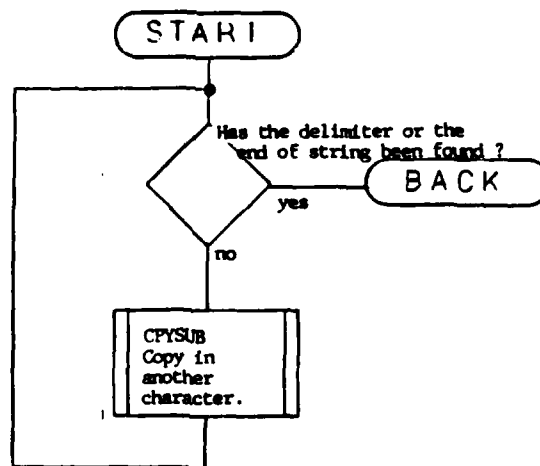
COPY STRING  
BETWEEN  
DELIMITER

PAGE 46



---

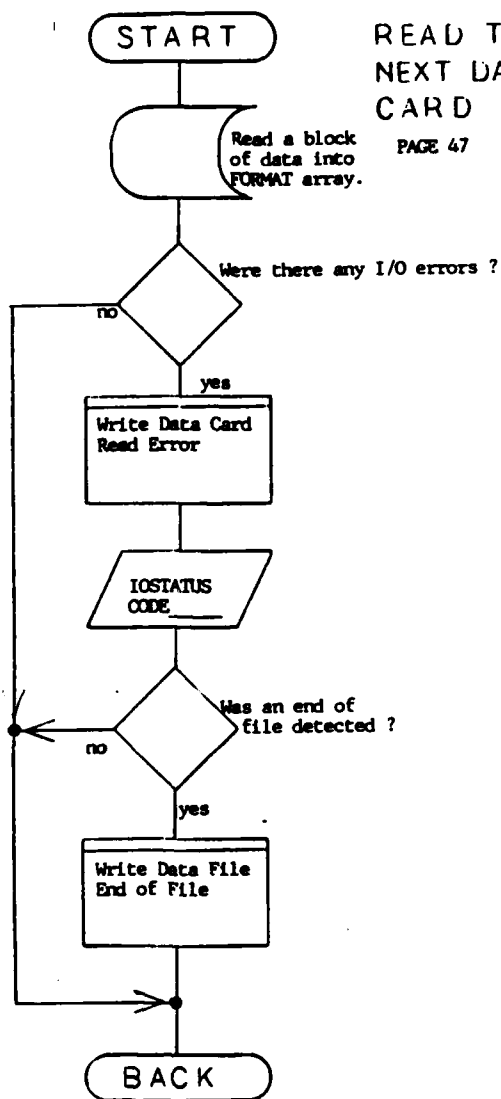
FIND THE NEXT  
DELIMITER



EDITREL

READ THE  
NEXT DATA  
CARD

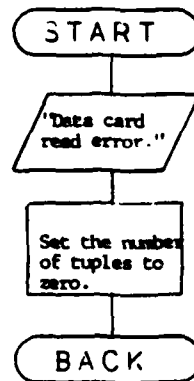
PAGE 47



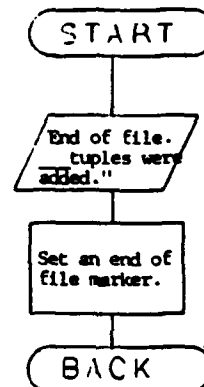
EDITREL

WRITE DATA  
CARD READ  
ERROR

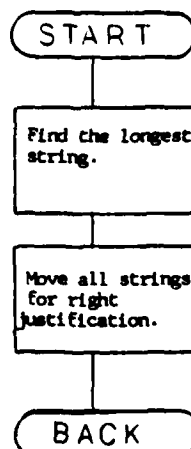
PAGE 48

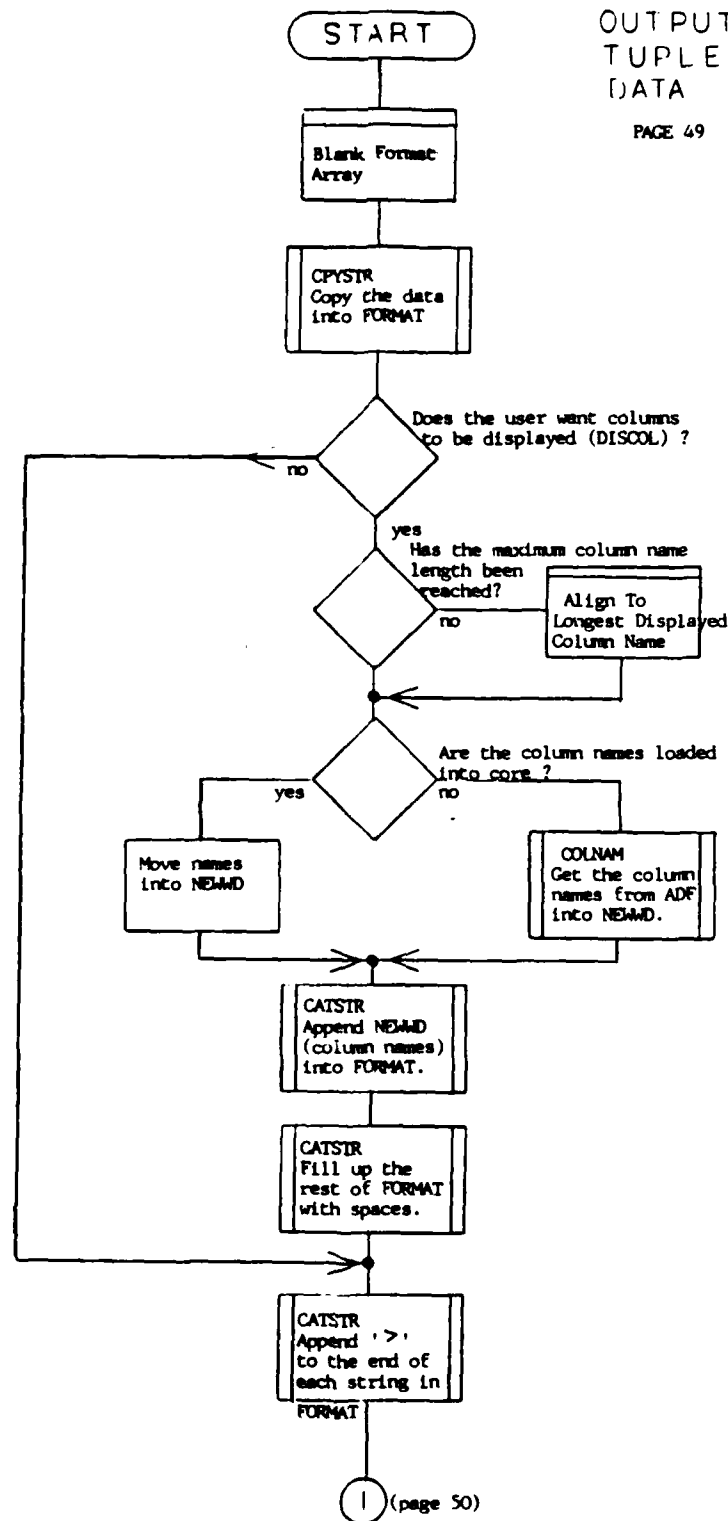


WRITE DATA  
FILE END  
OF FILE



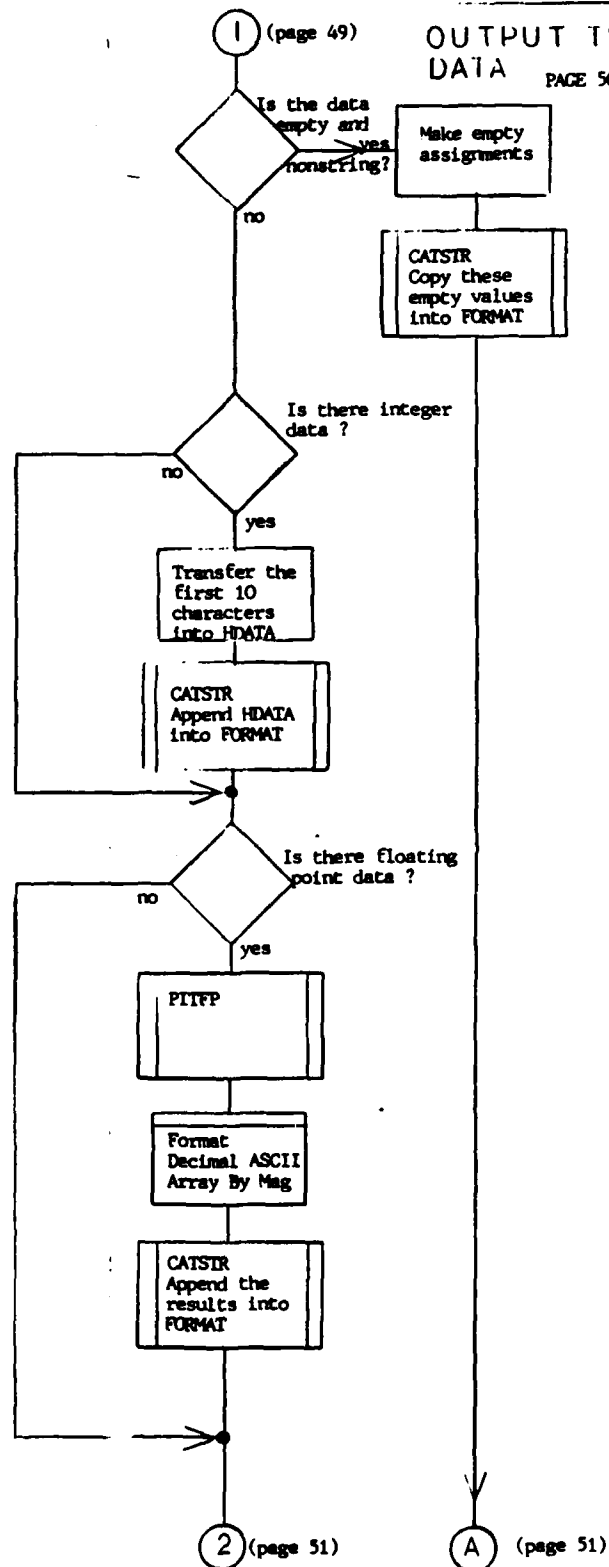
ASSIGN TO  
LONGEST  
DISPLAYED  
COLUMN  
NAME

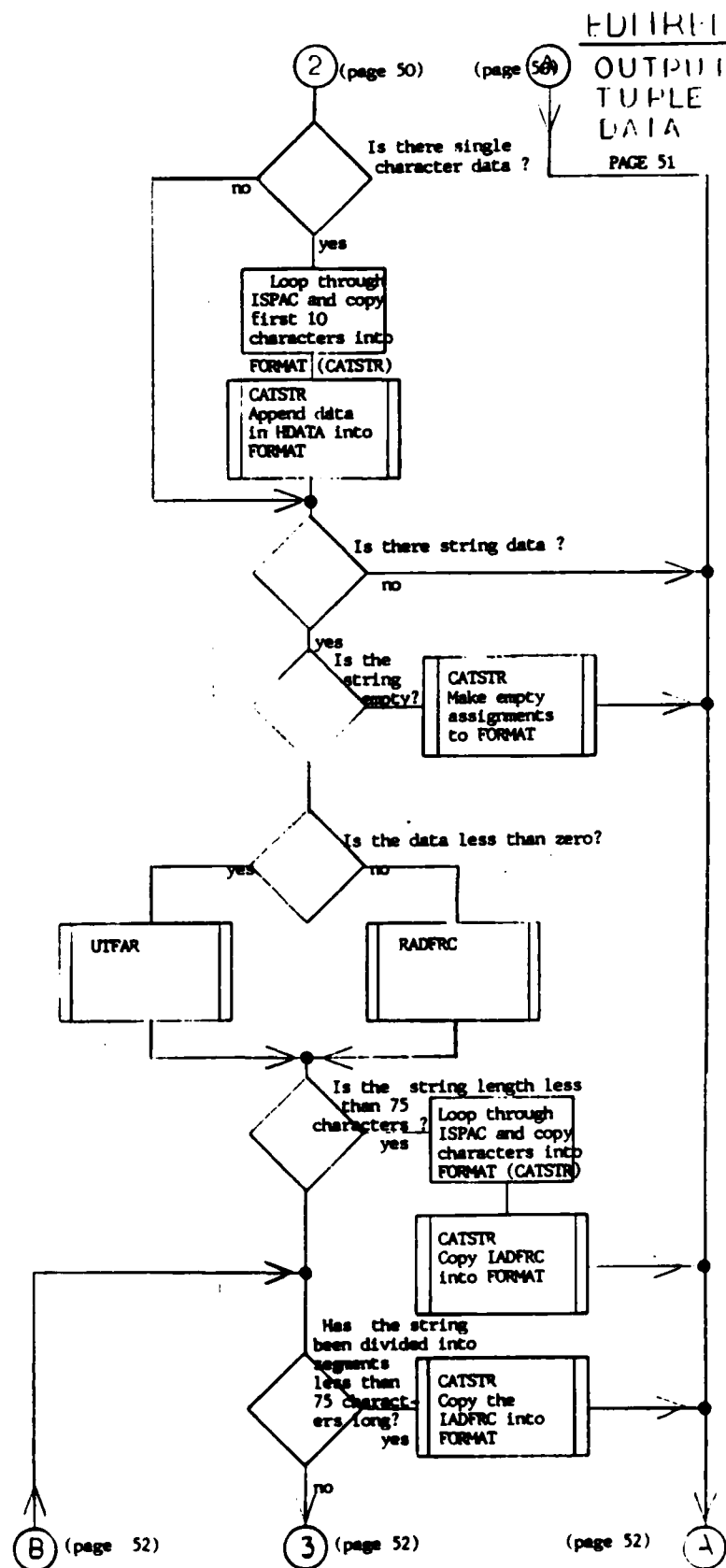




# EDITREL

OUTPUT TUPLE  
DATA PAGE 50



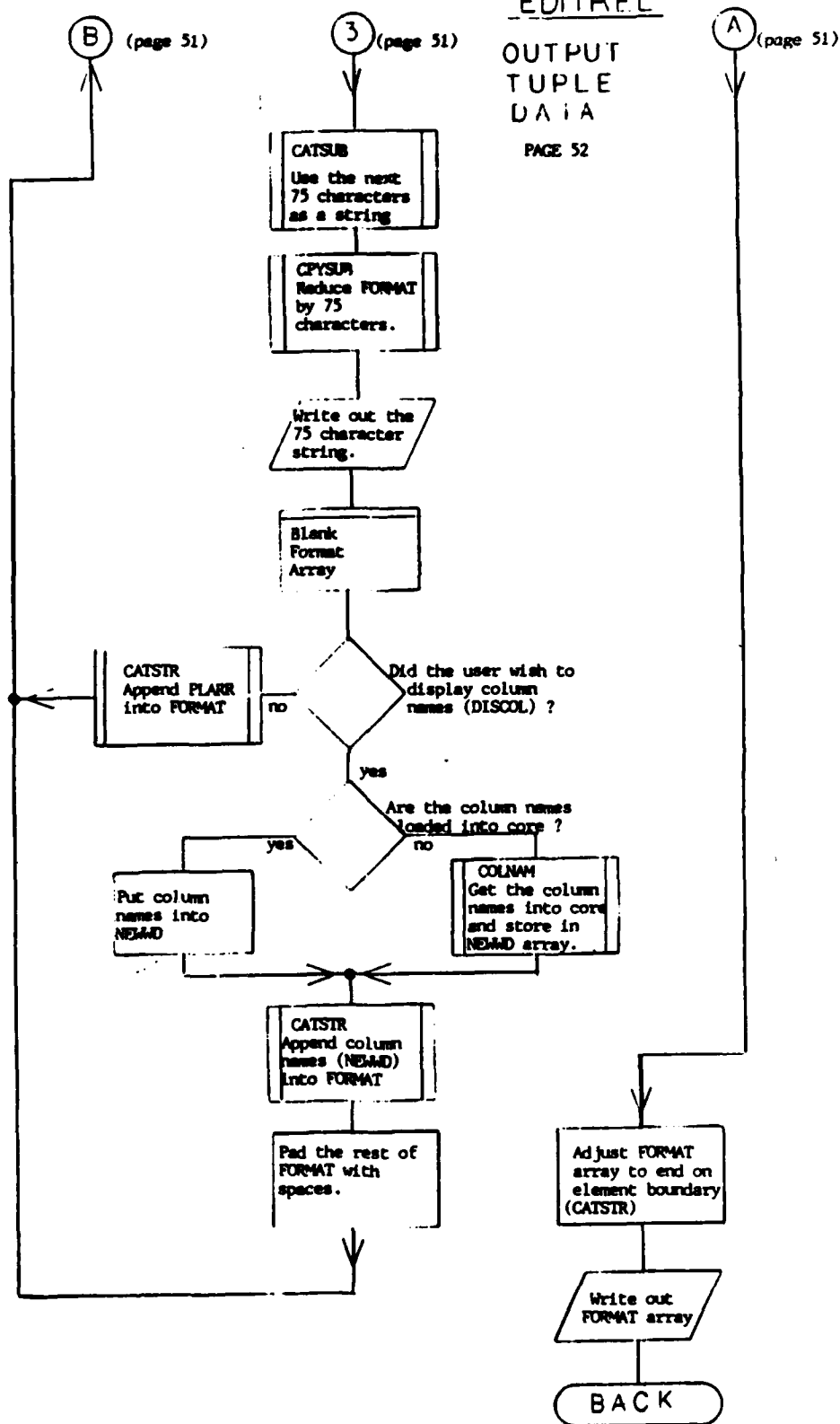




EDITREL

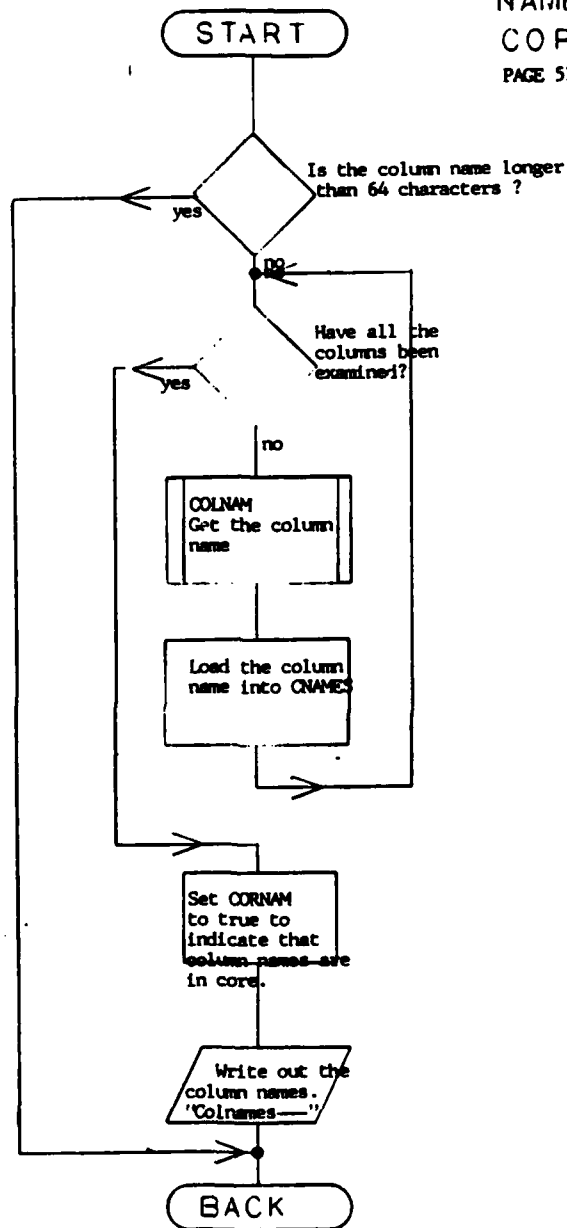
OUTPUT  
TUPLE  
DATA

PAGE 52



EDIT REL  
GET COLUMN  
NAMES INTO  
CORE

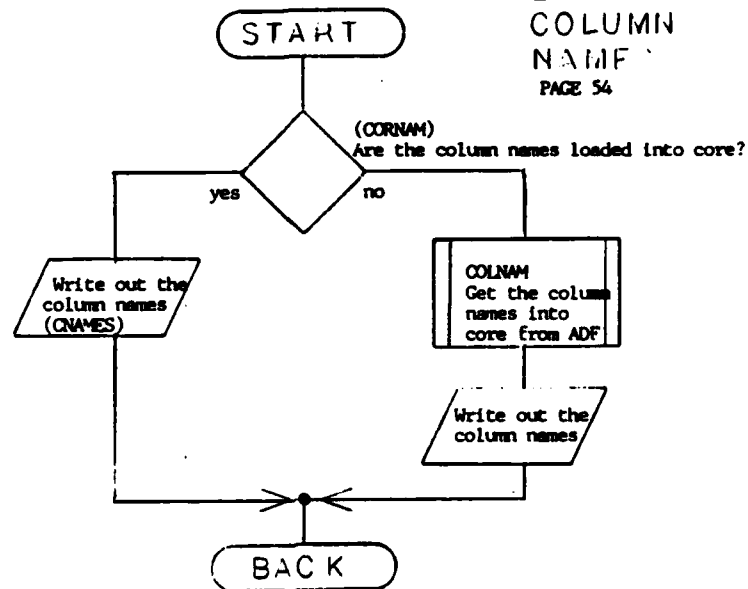
PAGE 53



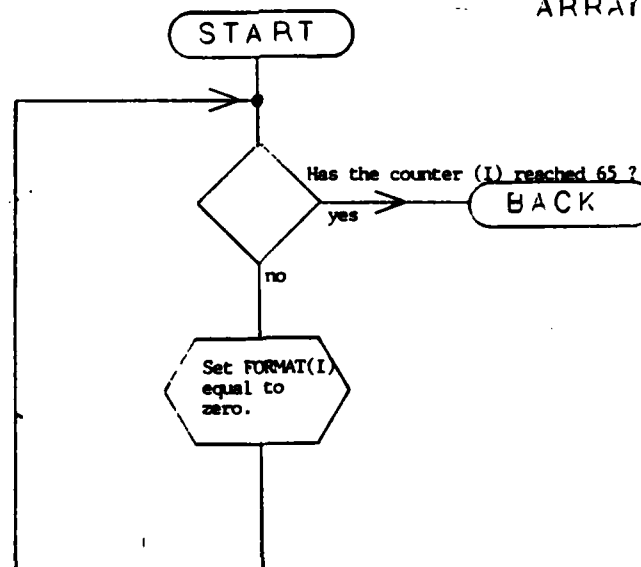
EDITREL

LIST  
COLUMN  
NAME

PAGE 54



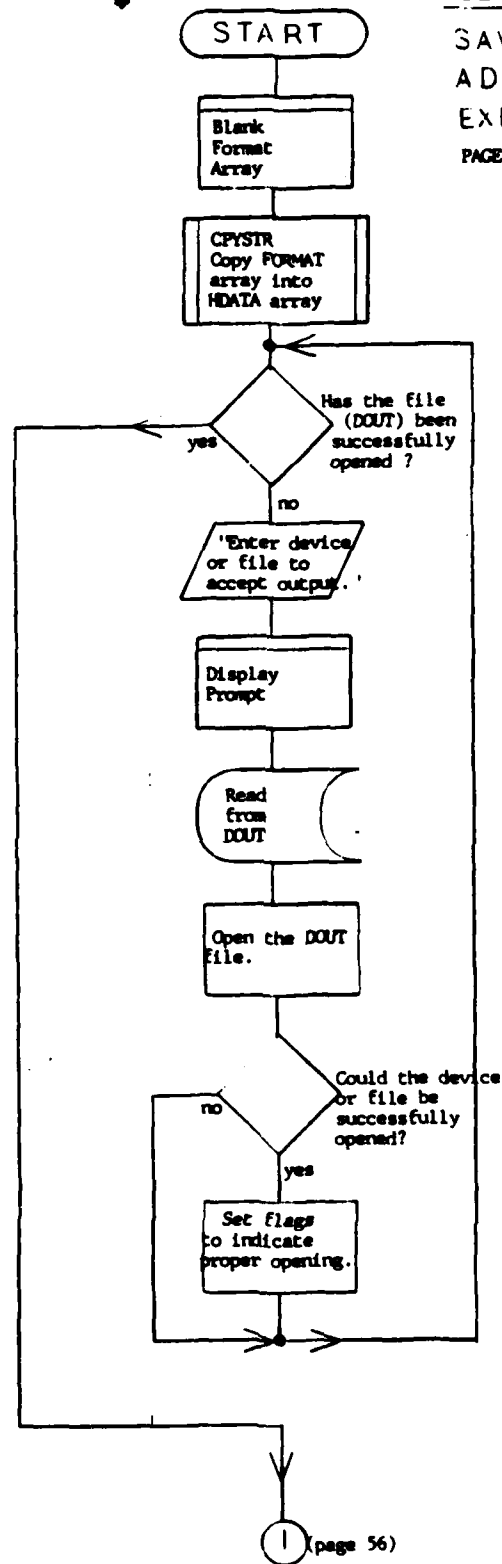
BLANK  
FORMAT  
ARRAY



EDITREL

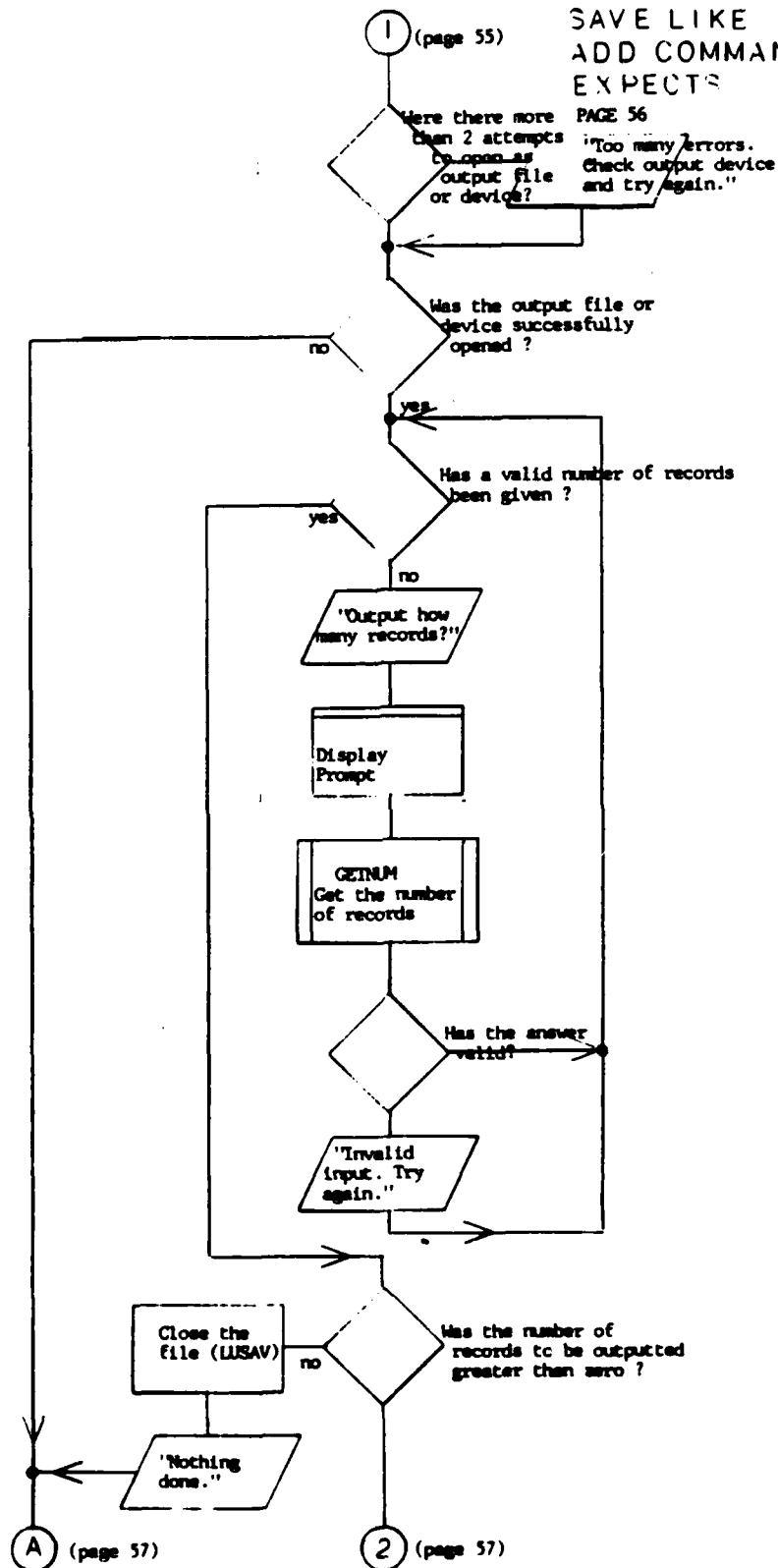
SAVE LIKE  
ADD COMMAND  
EXPECTS

PAGE 55



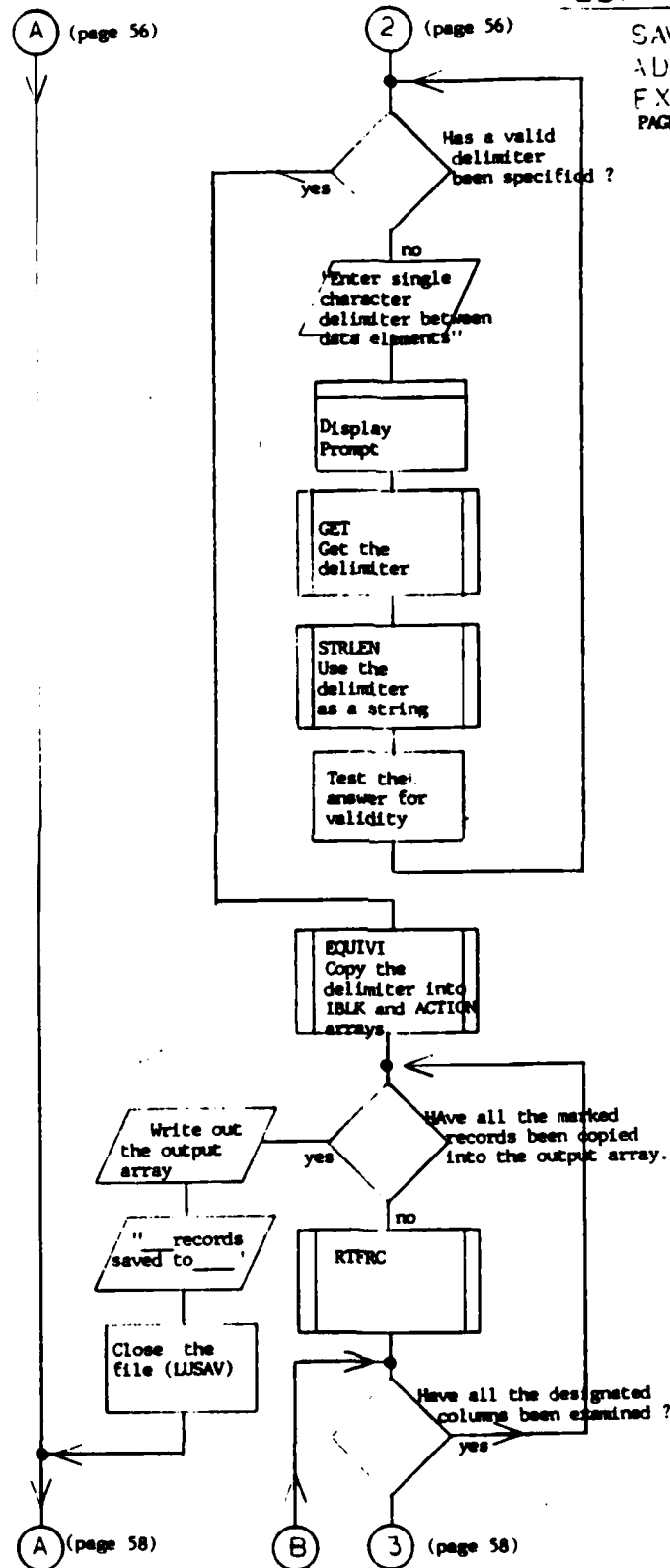
# EDITREL

SAVE LIKE  
ADD COMMAND  
EXPECTS



# EDITREL

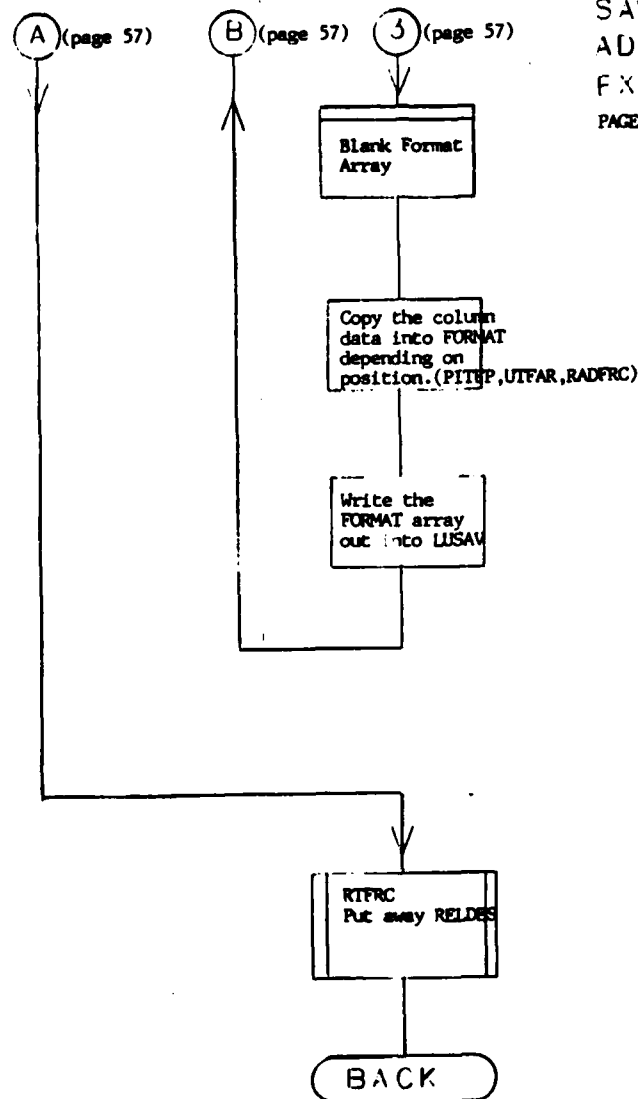
SAVE LIKE  
ADD CLIMATD  
EXPECTS  
PAGE 57



EDIT RFL

SAVE LIKE  
ADD COMMAND  
EXPECTS

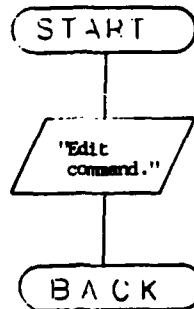
PAGE 58



## EDITREL

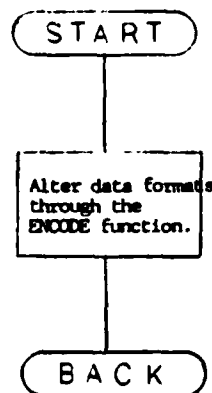
DISPLAY  
PROMPT

PAGE 59



---

FORMAT  
DECIMAL  
ASCII  
ARRAY  
BY MAG





APPENDIX H

NEWRELAT

OVERLAY: NEWRELAT.FLC

SUBROUTINE: NEWREL.FLC

SYNTAX:

CALL

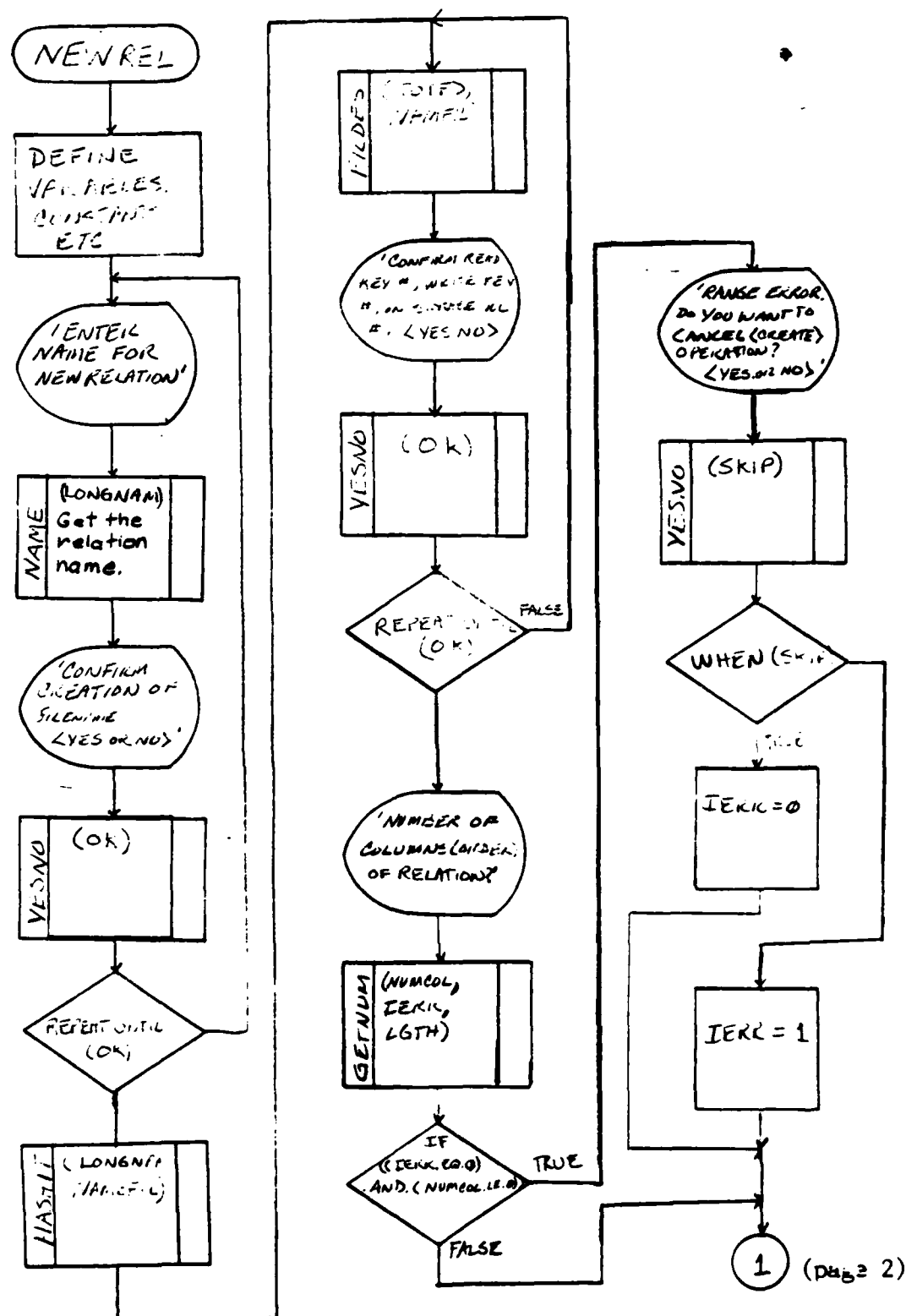
NEWREL(IDENT,ISAFTE,ITDF,INC,INR,IDEM,IDADF,ITDF,ITFRC,IADFRC)

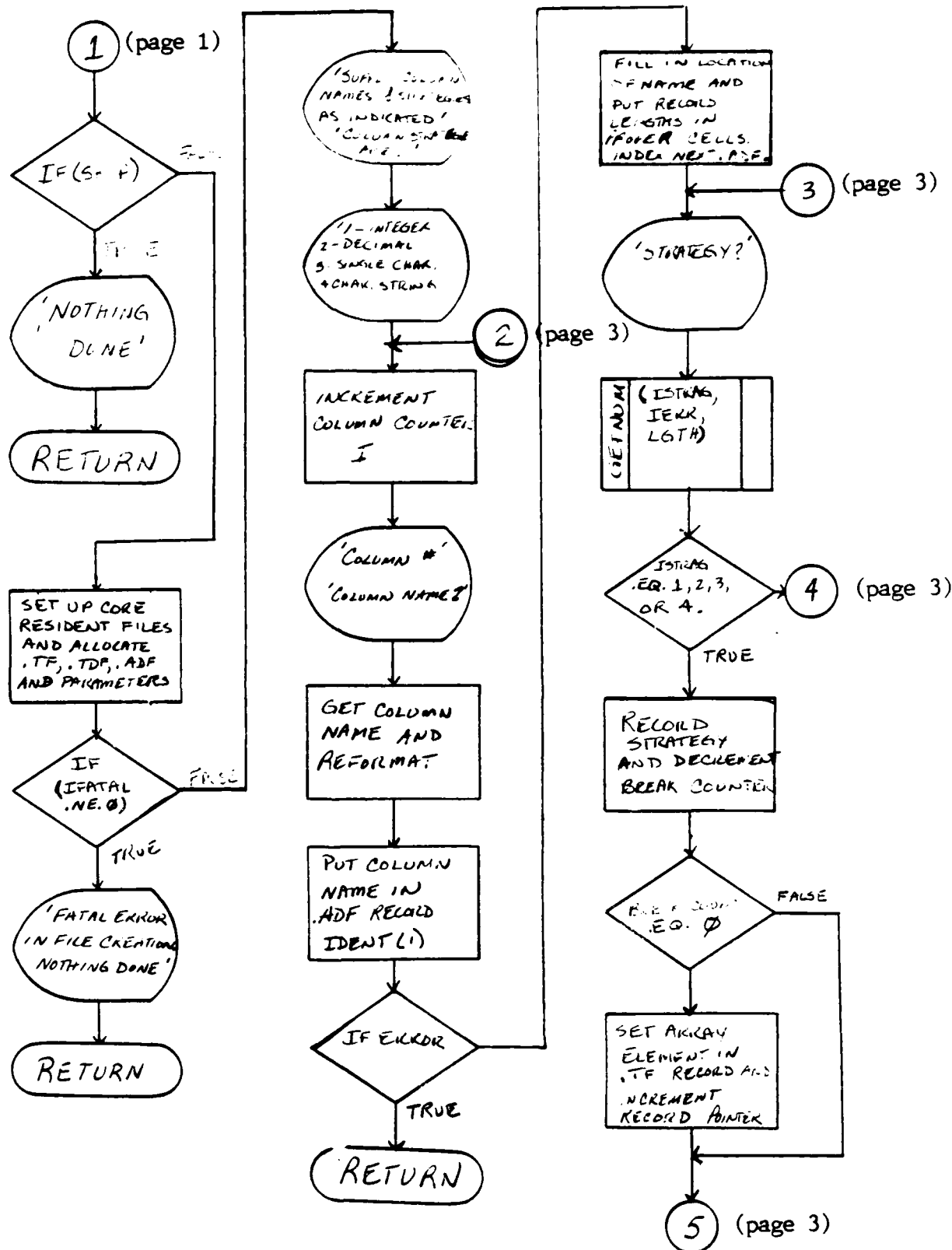
NEWREL---CREATE A NEW RELATION.

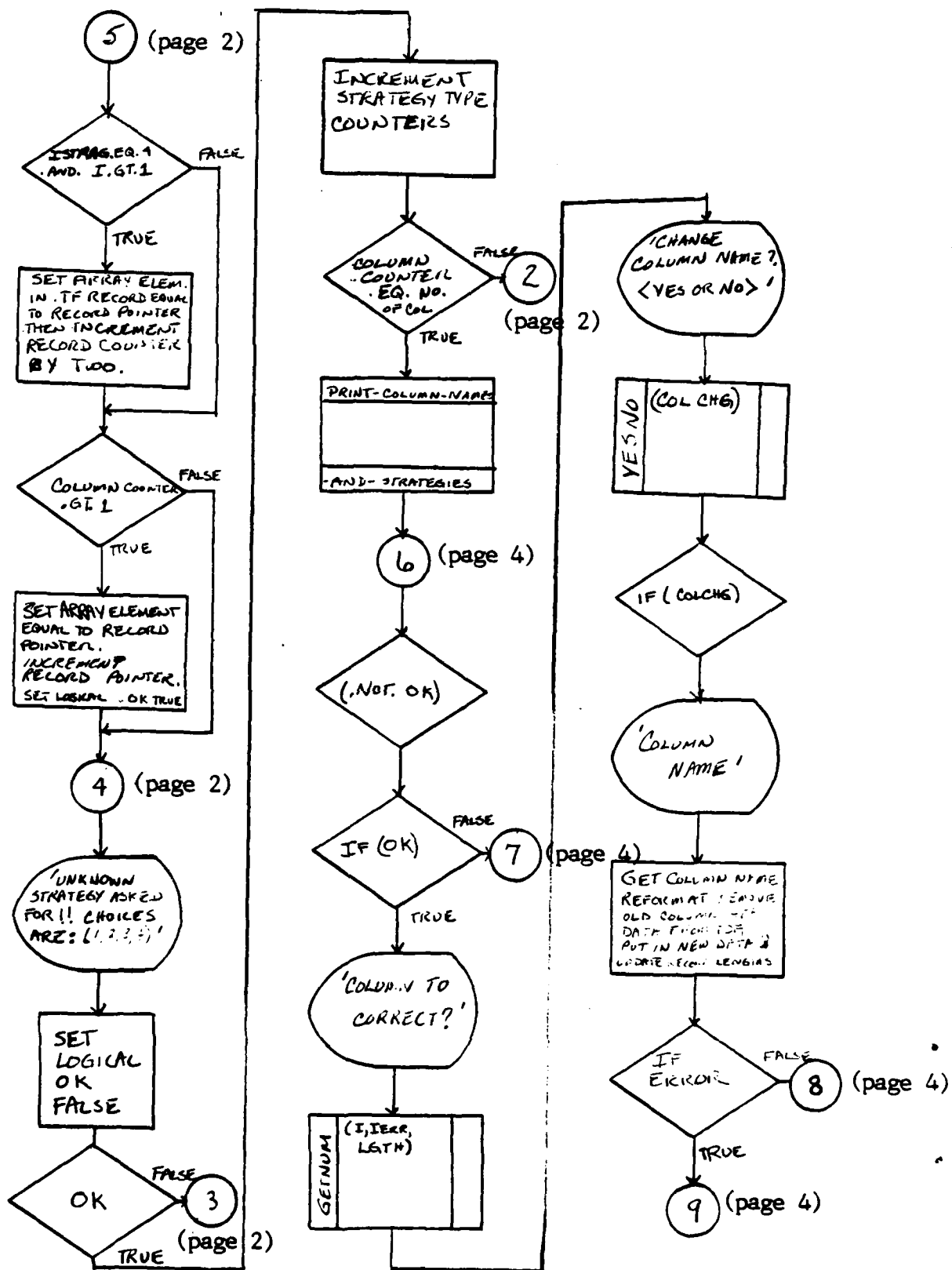
THIS ROUTINE IS USED TO INITIALLY ESTABLISH A RELATION. THE THREE FILES .TF, .TDF AND .ADF ARE ALLOCATED, AND THE COLUMN HEADINGS AND STRATEGIES ARE REQUESTED AND SET UP. ALSO THE INITIAL VALUES FOR THE RELATION (IDENT AND ITDF) ARE ASSIGNED TO THE FILES.

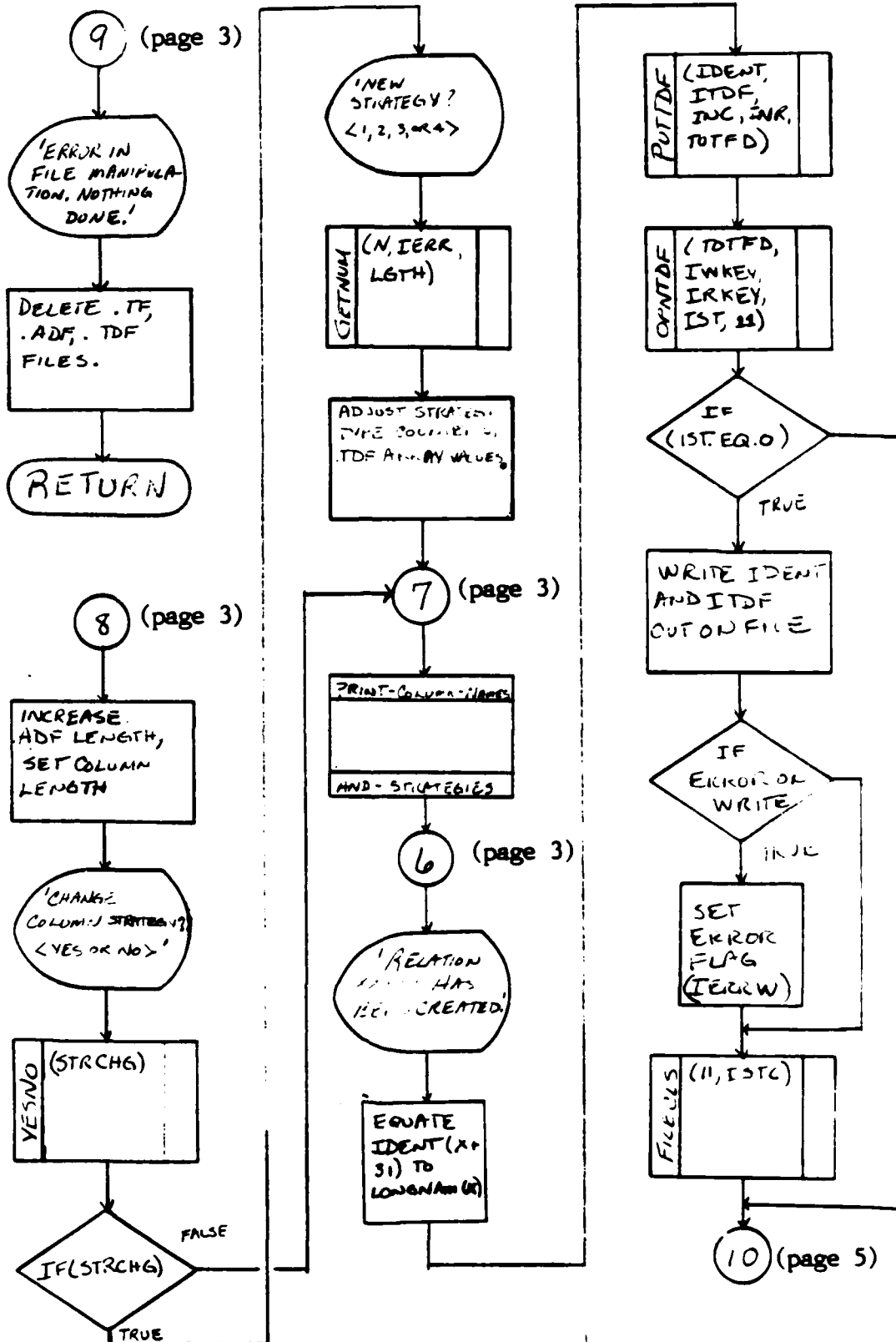
THE SUBROUTINE ARGUMENTS ARE:

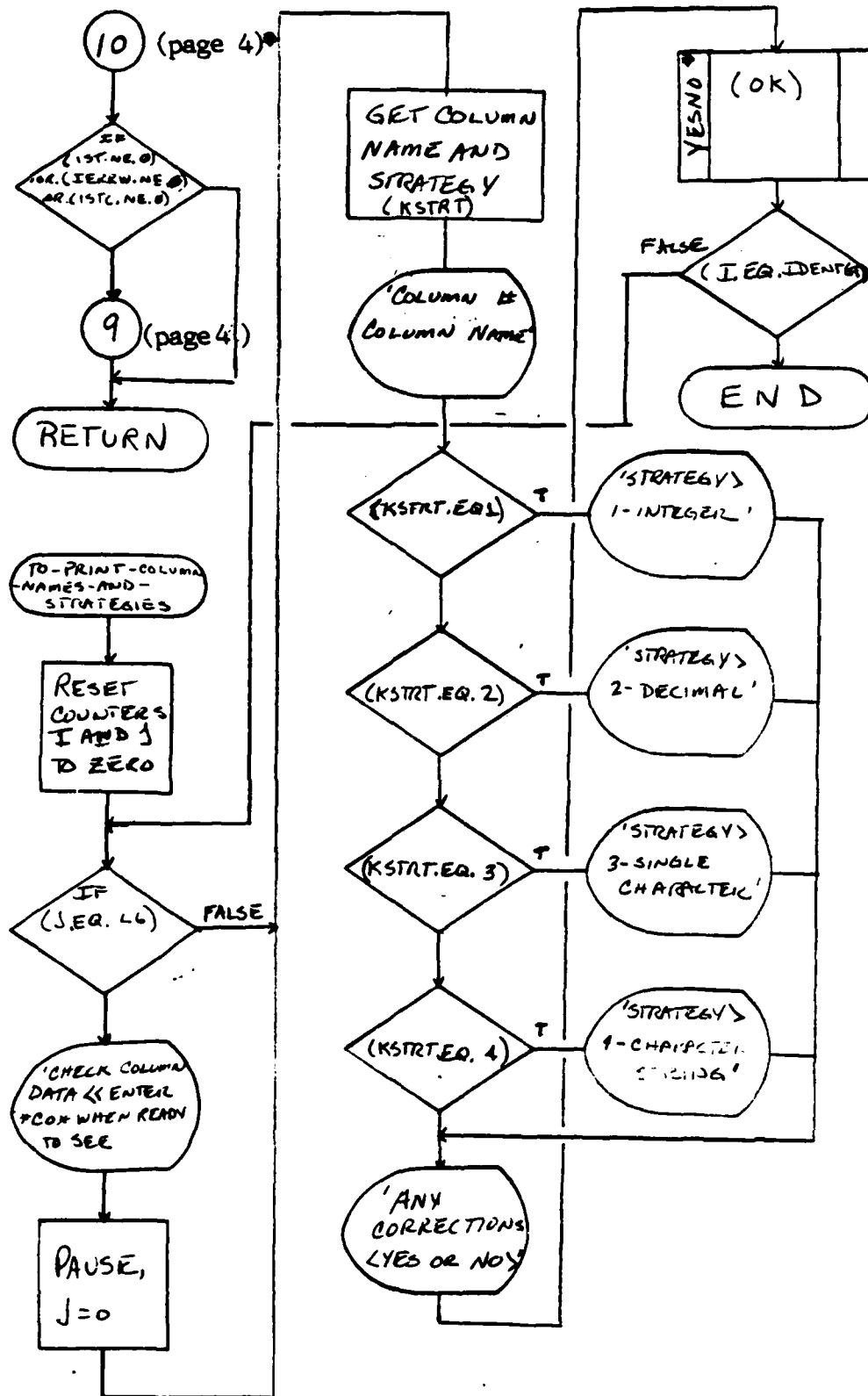
1. IDENT - RELATION STATUS; CORE RESIDENT,
2. IDEM - DIMENSION OF IDENT,
3. ITDF - CORE RESIDENT TDF,
4. INC - NUMBER OF COLUMNS IN TDF, AND
5. INR - NUMBER OF RECORDS IN TDF.











**APPENDIX I**

**DELREL**



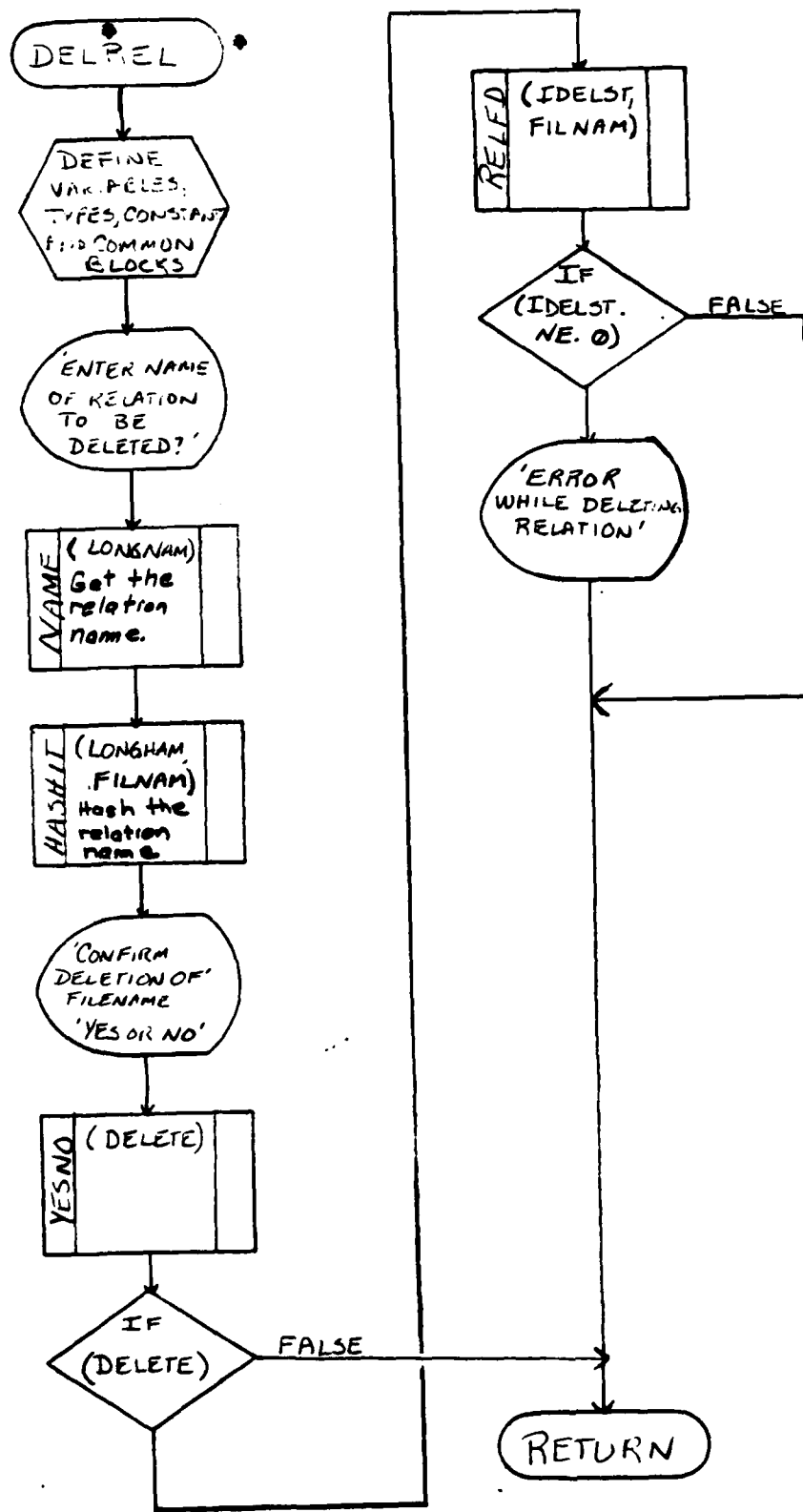
OVERLAY: DELETREL.FLC

SUBROUTINE: DELREL.FLC

SYNTAX:

CALL DELREL

THIS ROUTINE DELETES AN UNWANTED RELATION. ALL FILES .TF, .TDF. AND .ADF ARE DELETED. THIS ROUTINE REQUIRES CONFIRMATION THAT THE RELATION NAMED IS THE CORRECT RELATION TO BE DELETED BEFORE IT IS DELETED.



## APPENDIX J

MODCOL

**Subroutine: MODCOLUM**

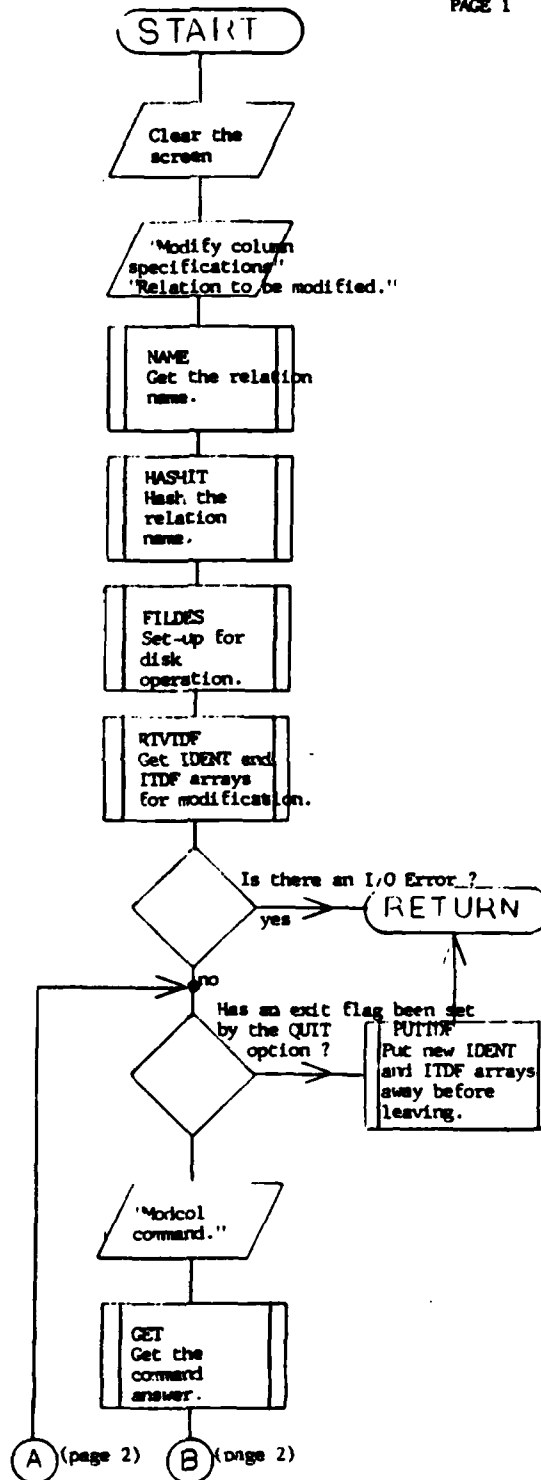
- This subroutine provides for the manipulation of columns in an existing relation. No changes in strategy are allowed, since this would assume a change in data.

The basic functions provided are:

- (1.) Delete a column
- (2.) Add a column
- (3.) Alter a column name
- (4.) List current column names

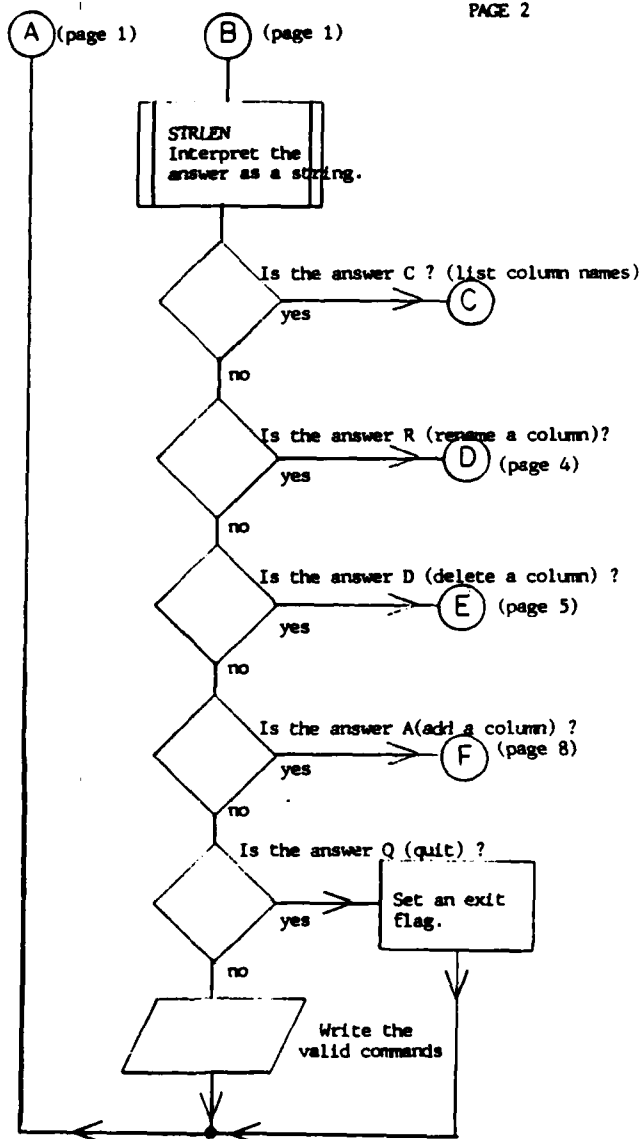
# MODCOLUM

PAGE 1



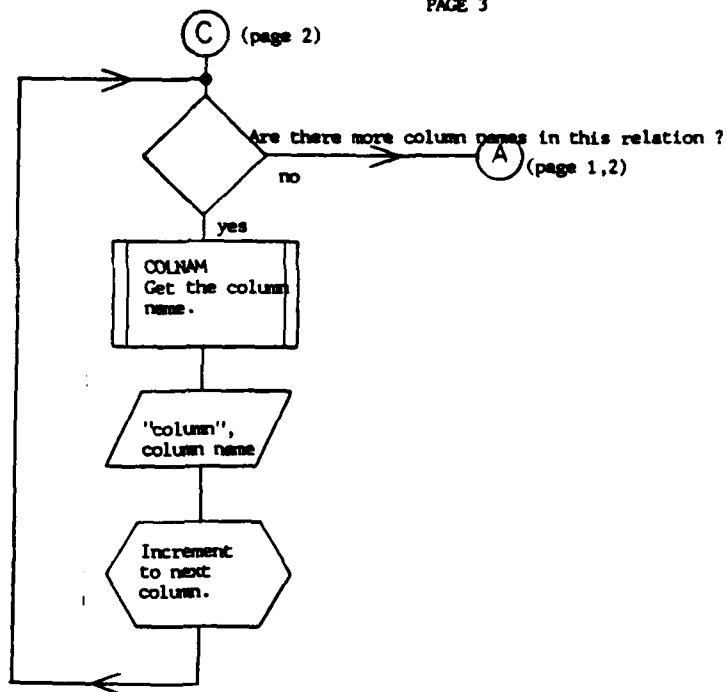
# MODCOLUM

PAGE 2



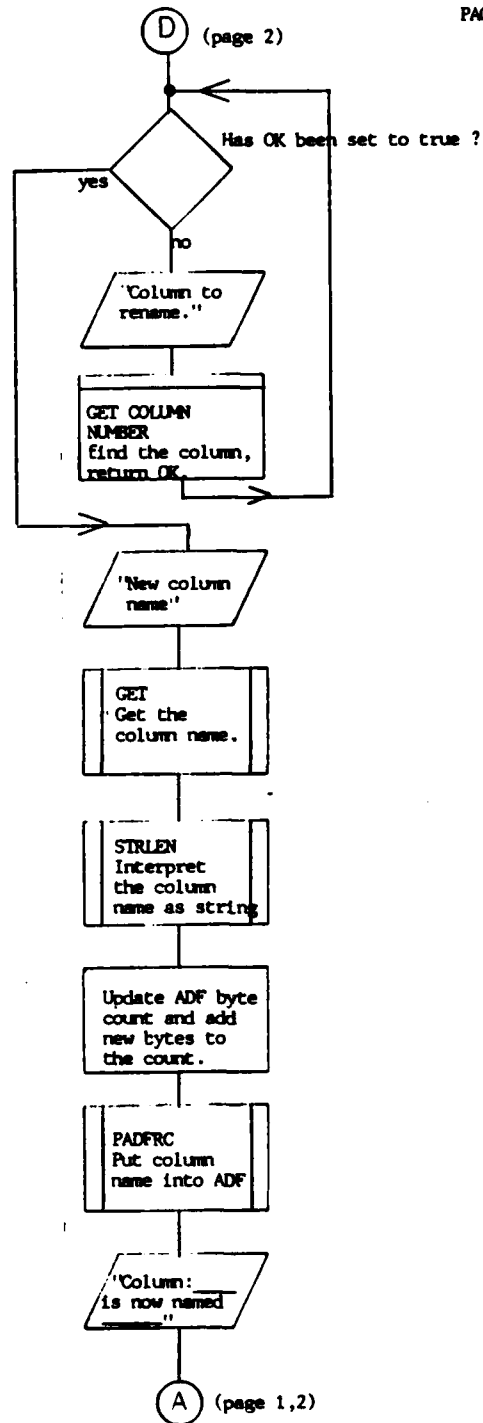
# MODCOLUM

PAGE 3



# MODCOLUM

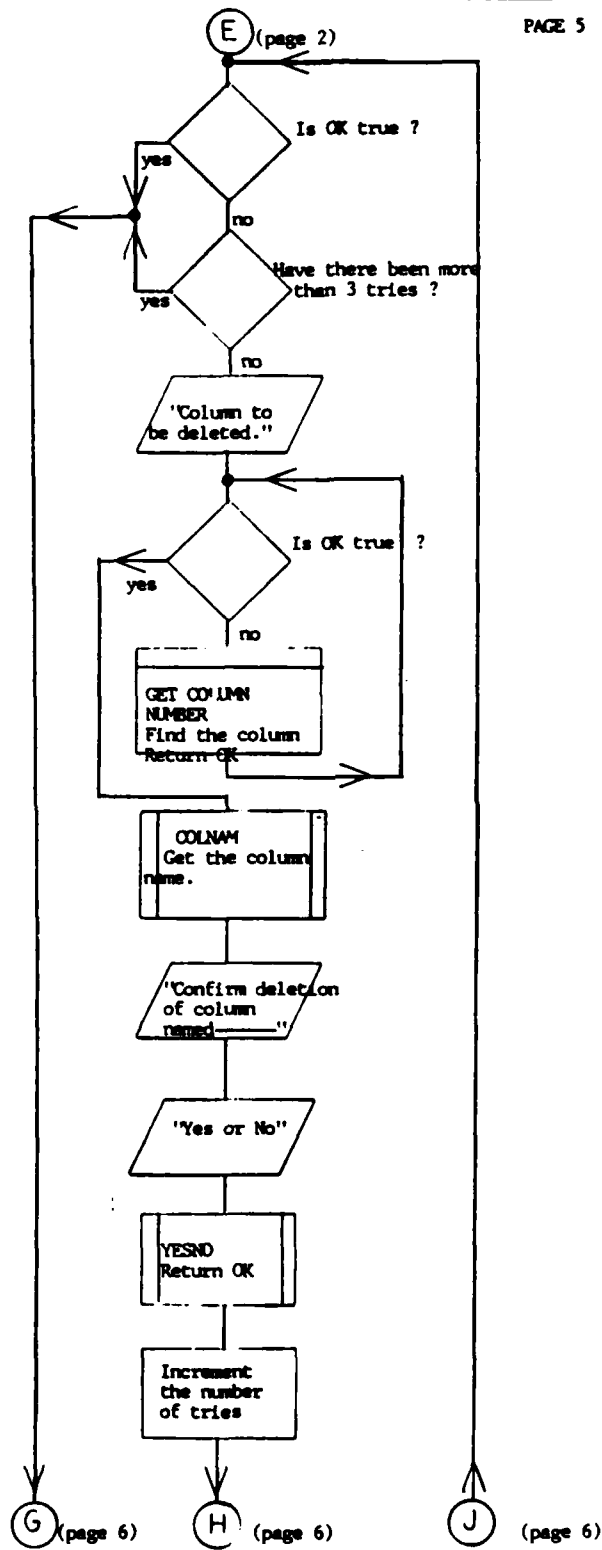
PAGE 4





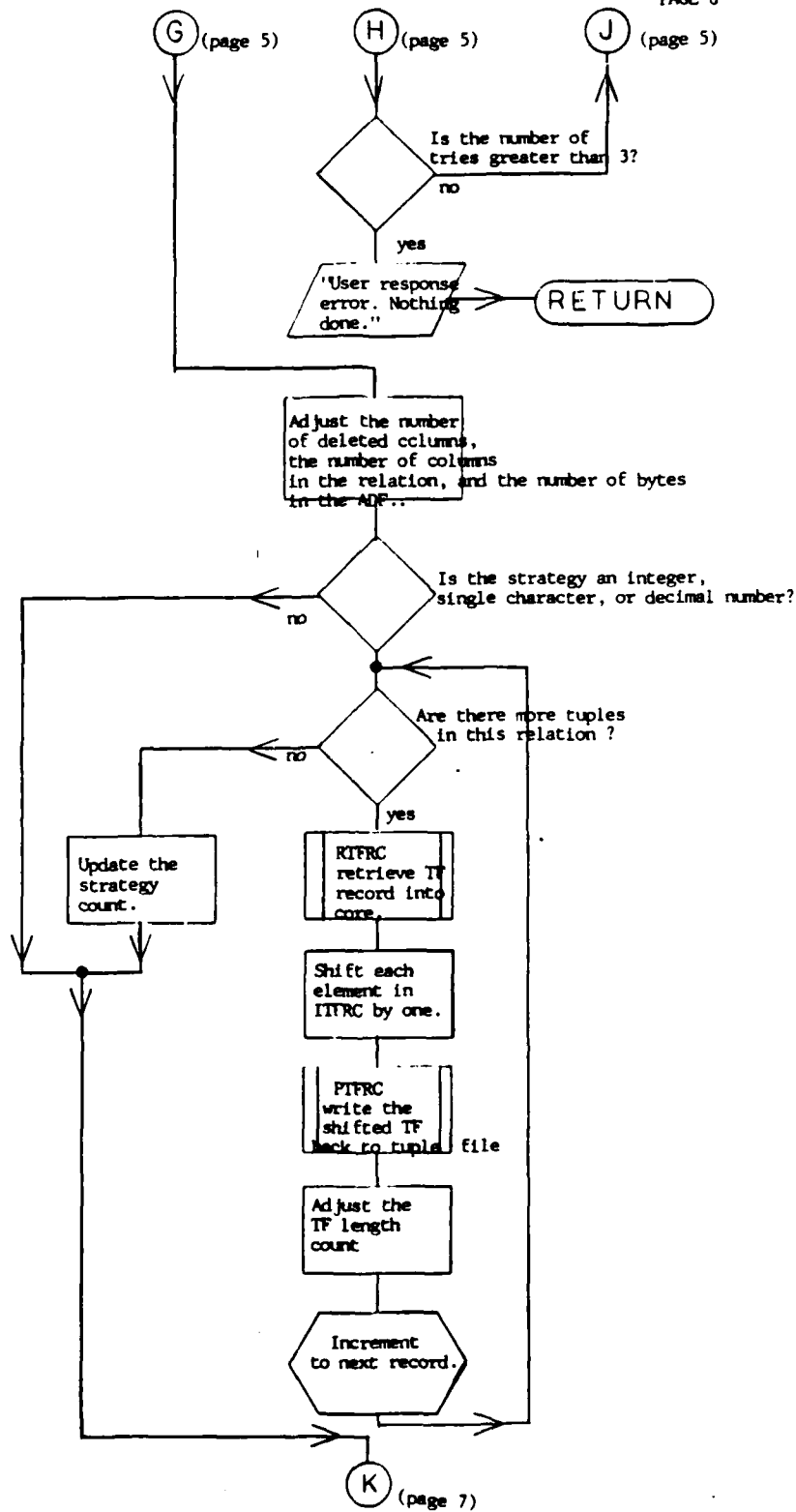
# MODCOLUM

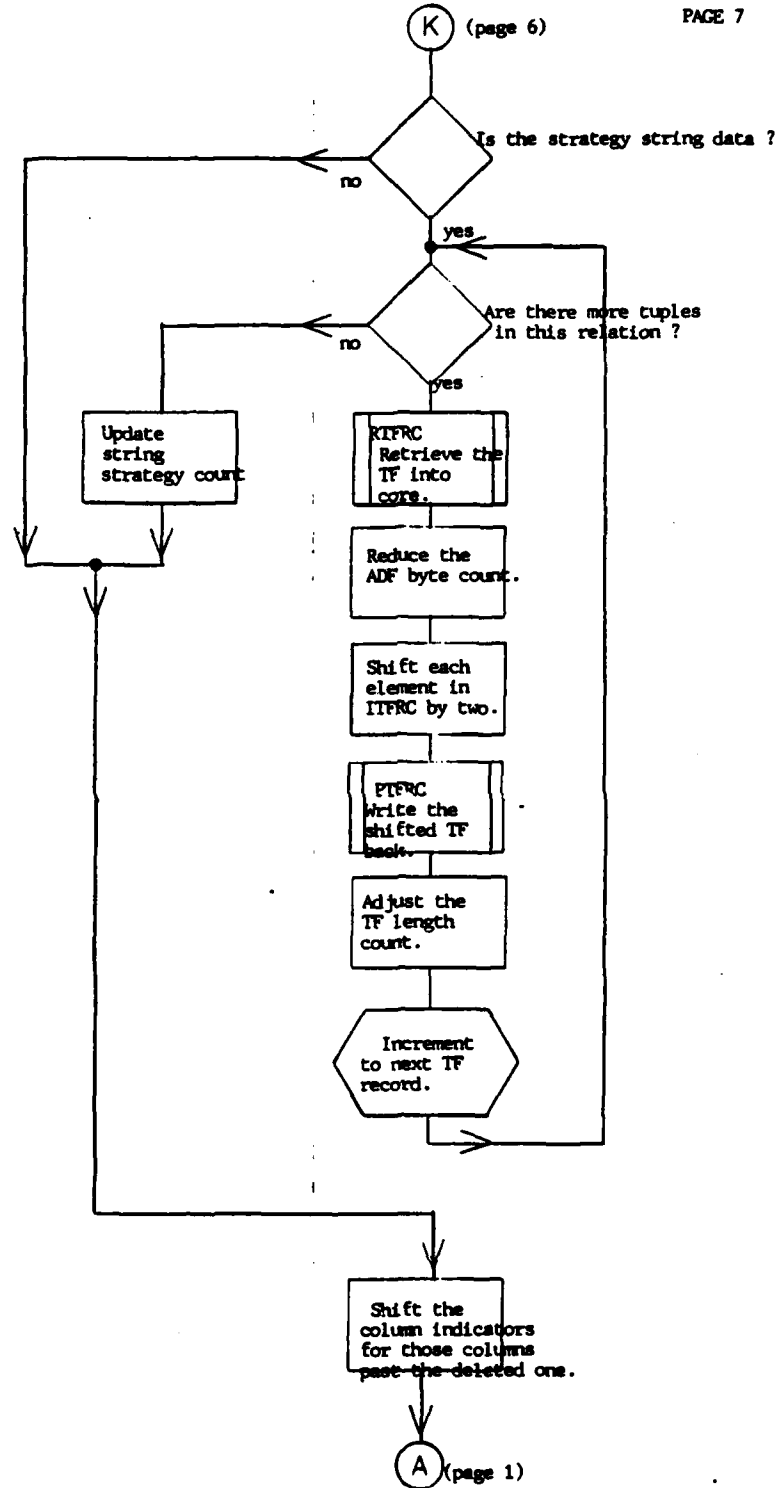
PAGE 5



# MODCOLUM

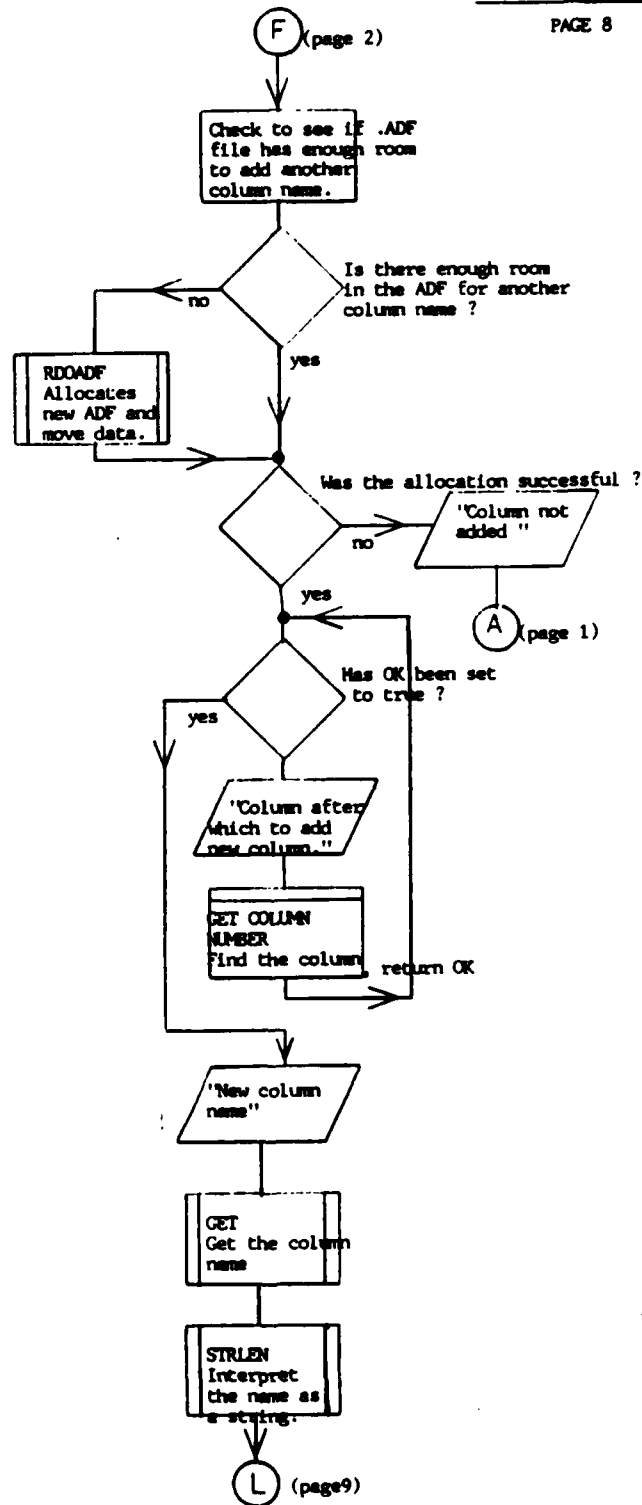
PAGE 6

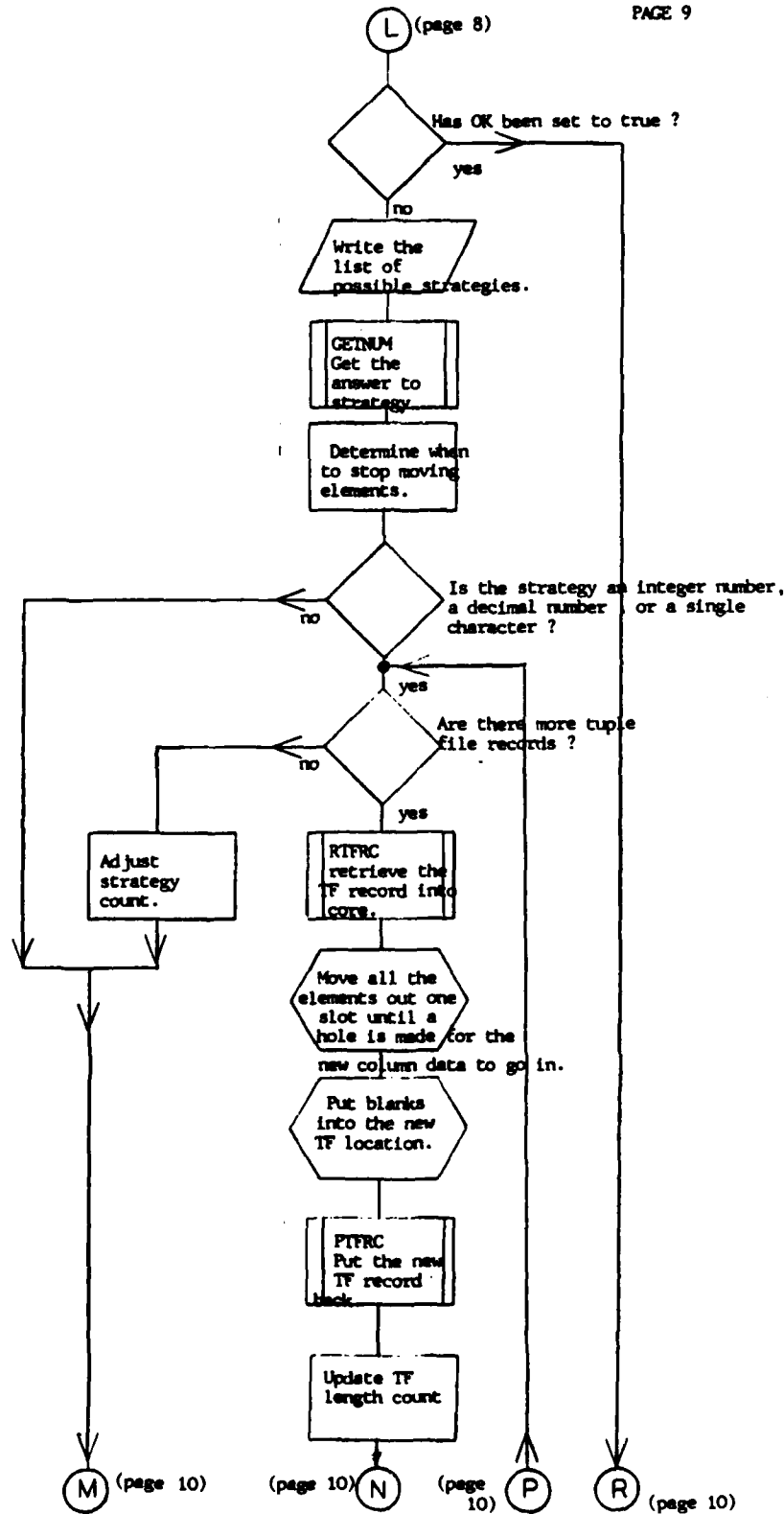




# MODCOLUM

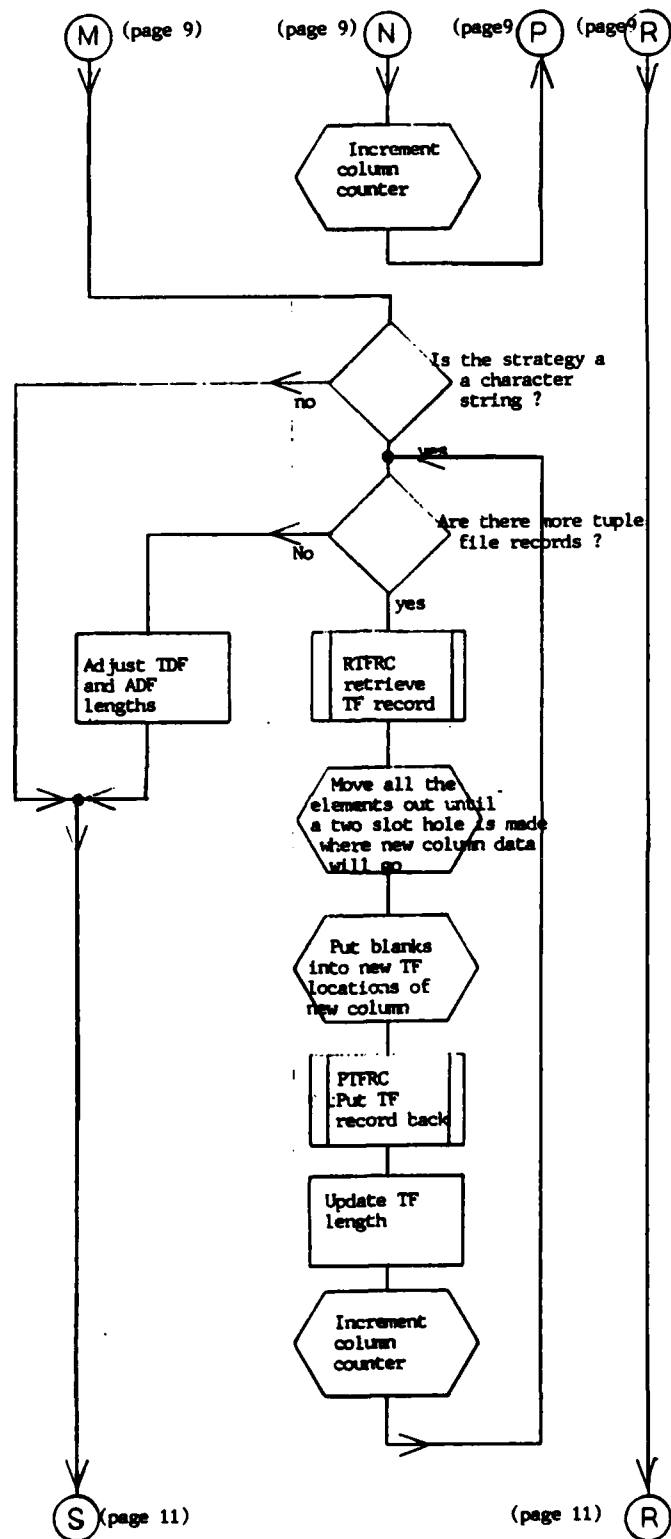
PAGE 8





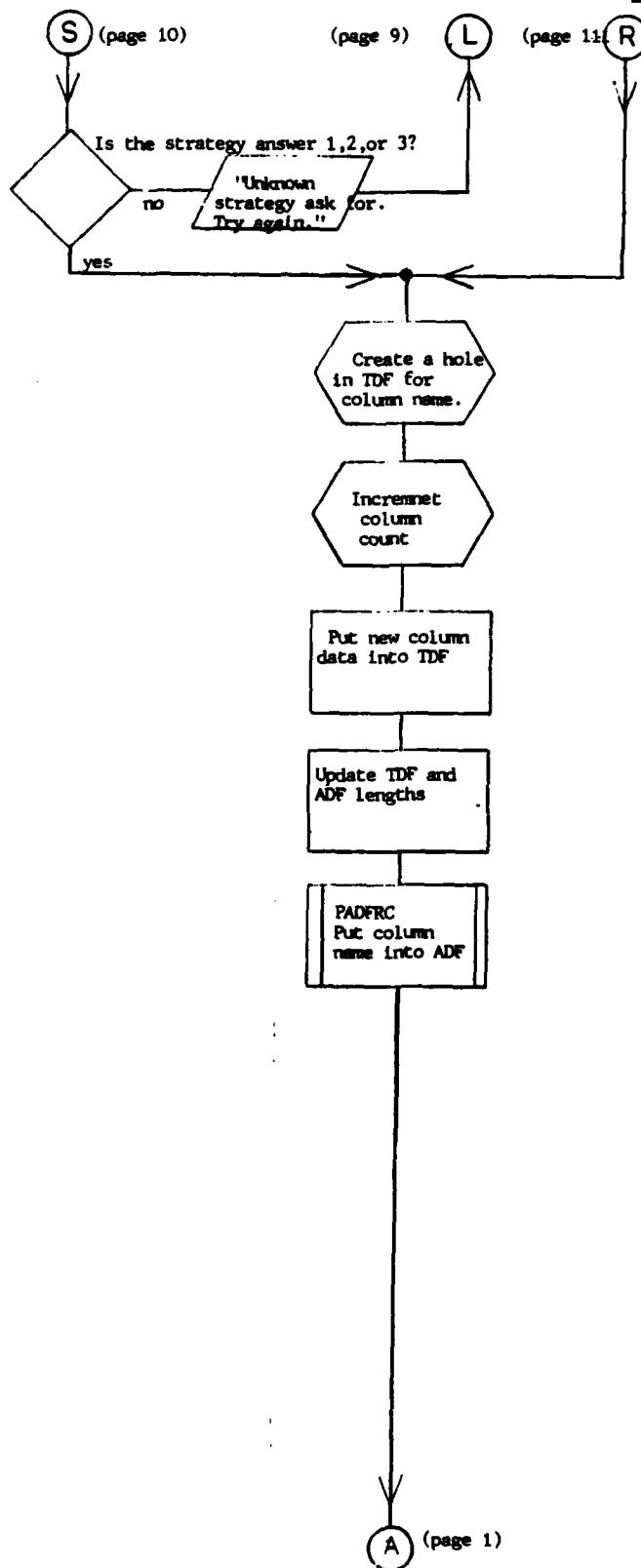
# MODCOLUM

PAGE 10

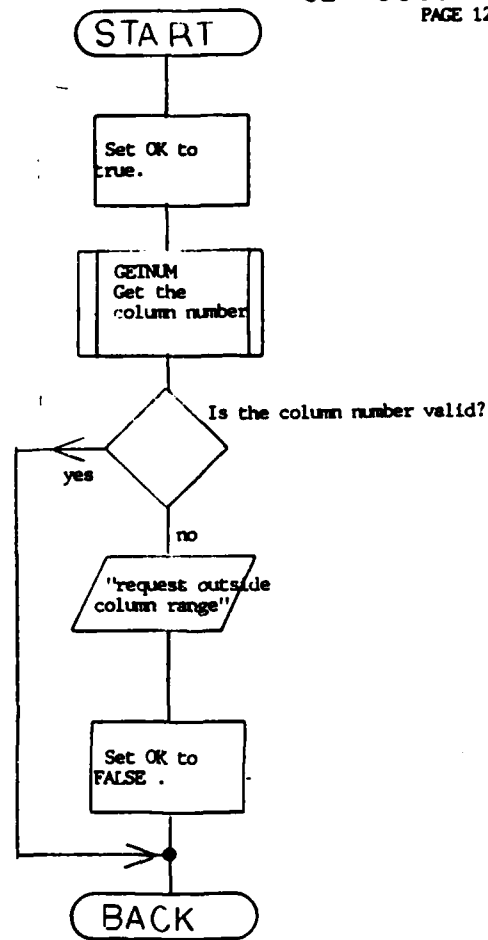


# MODCOLUM

PAGE 11



MODCOLUM  
GET COLUMN NUMBER  
PAGE 12





**APPENDIX K**

**MERGRL**

Subroutine: MERGEREL

This subroutine allows the user to merge portions of two similar relations into a third composite relation.

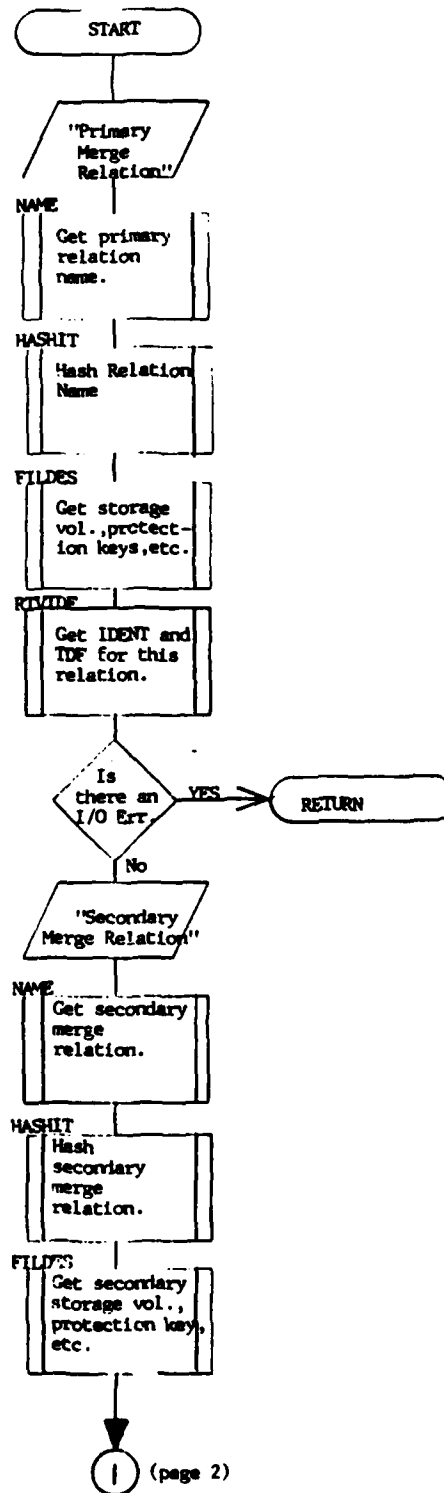
The options are:

(1.) Merge two relations into a third relation containing only those records which meet the matching of like column pairs.

(2.) Add two relations by copying first into a third relation and then appending the second relation. The column headings must be identical to use this option.

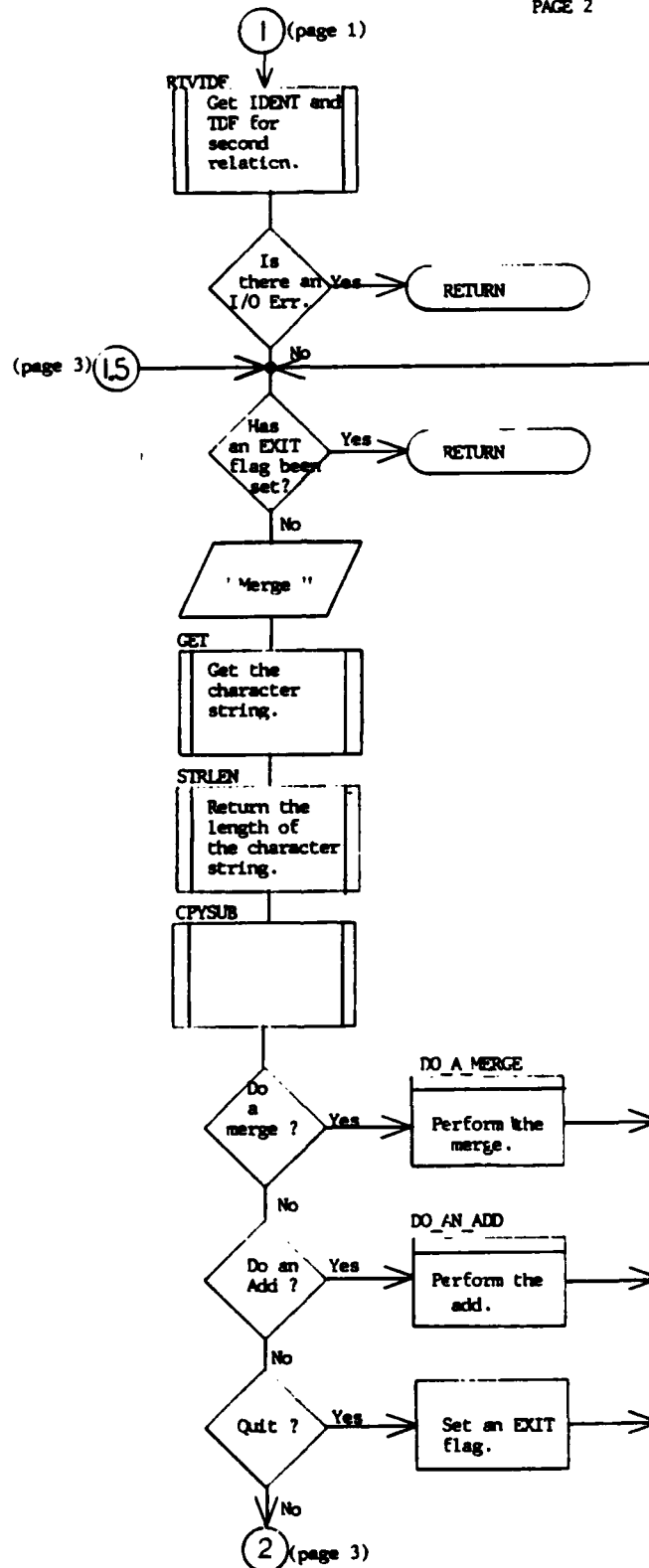
# MERGEREL

PAGE 1



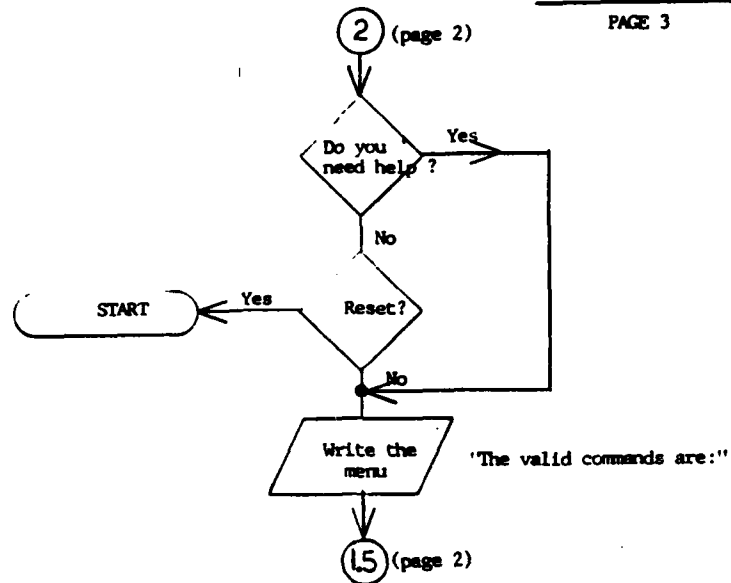
# MERGEREL

PAGE 2

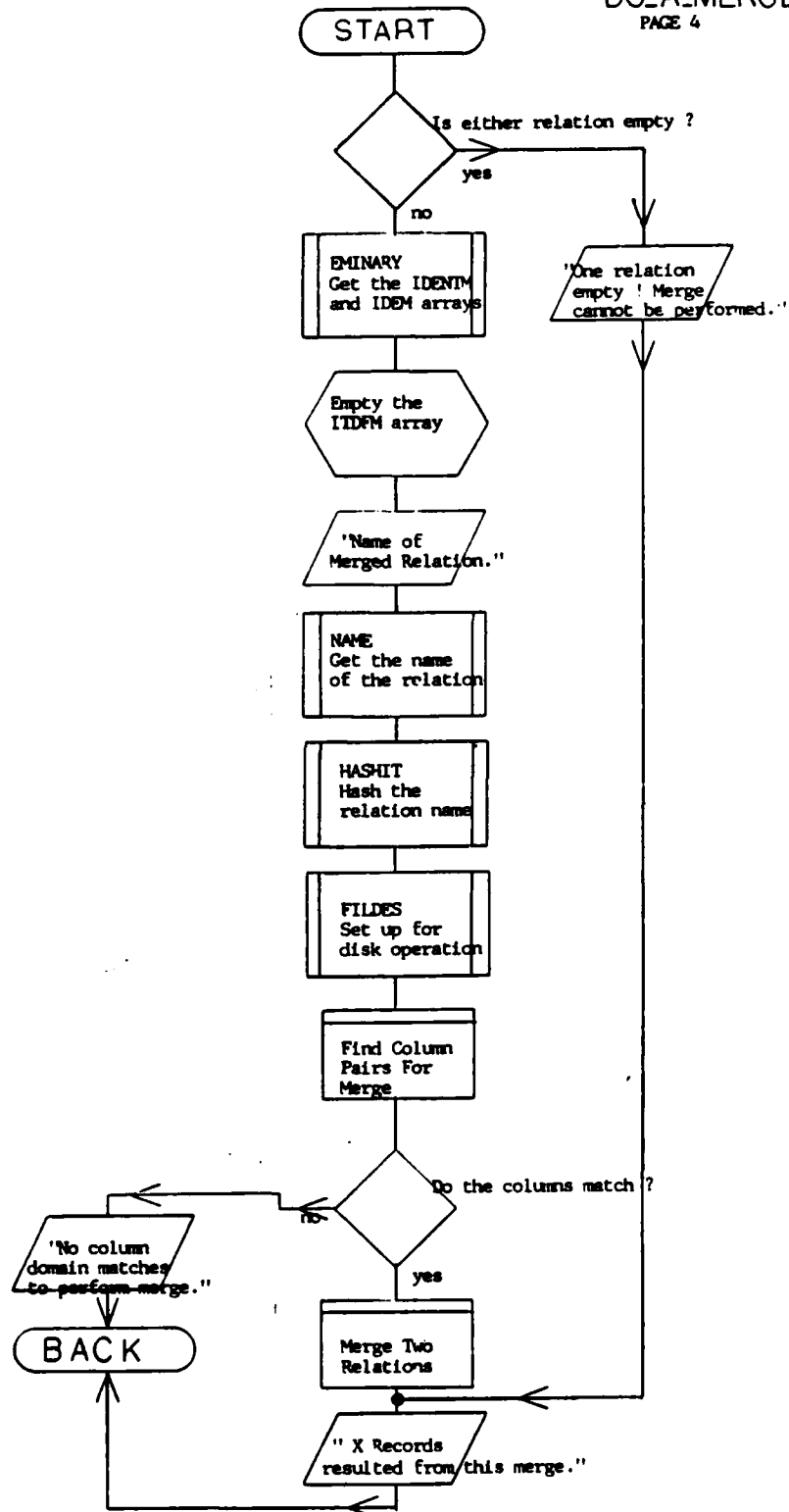


# MERGEREL

PAGE 3



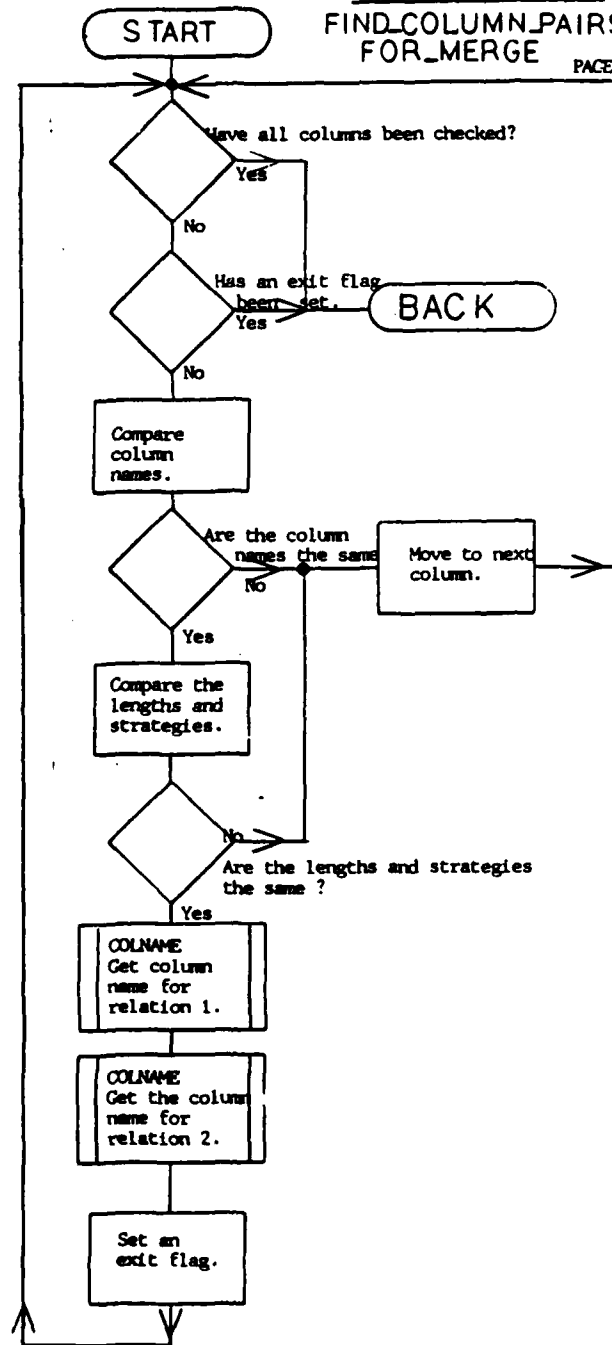
MERGEREL  
DO\_A\_MERGE  
PAGE 4



# MERGEREL

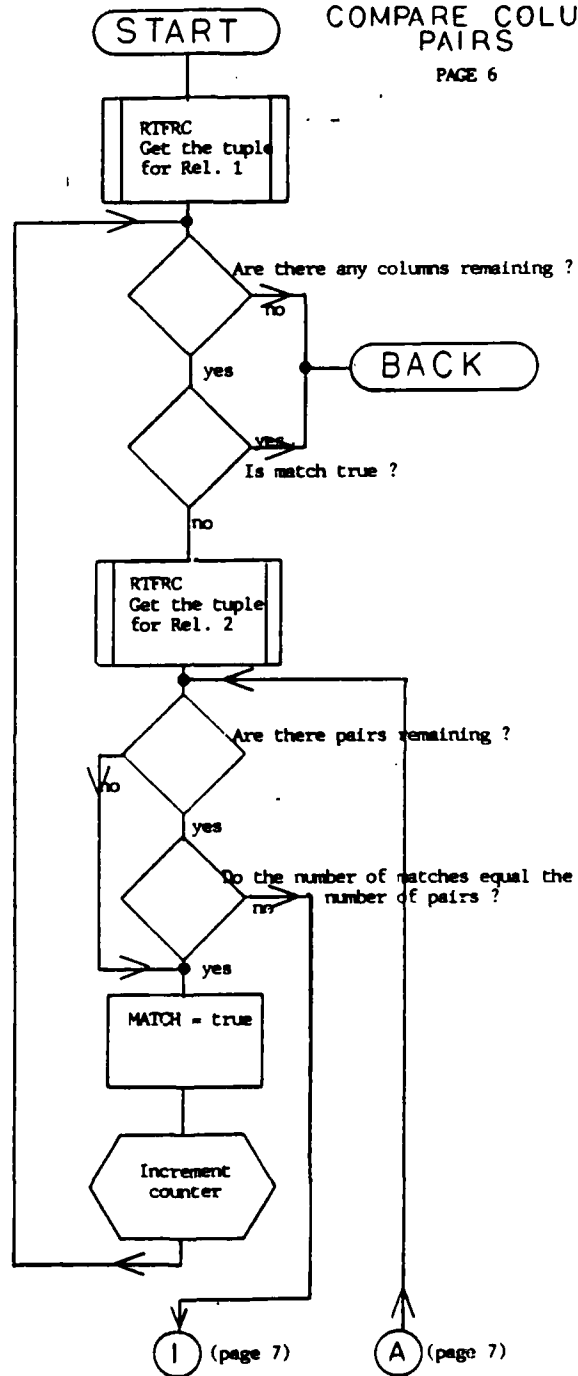
## FIND\_COLUMN\_PAIRS- FOR\_MERGE

PAGE 5



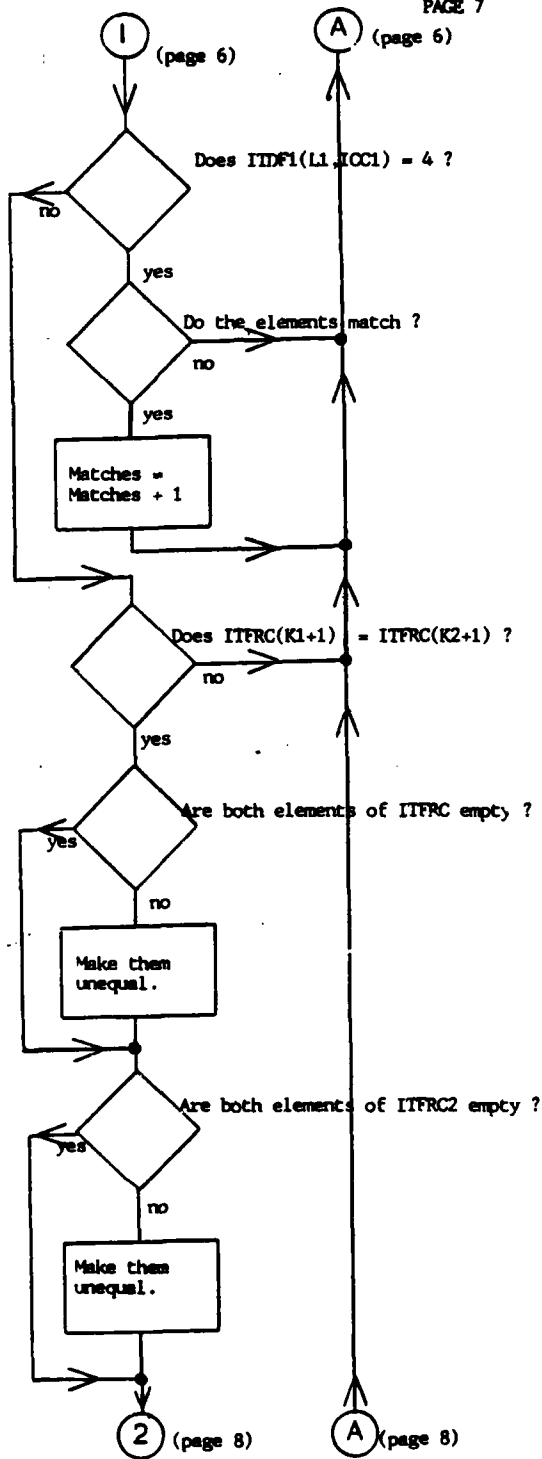
MERGEREL  
COMPARE COLUMN  
PAIRS

PAGE 6

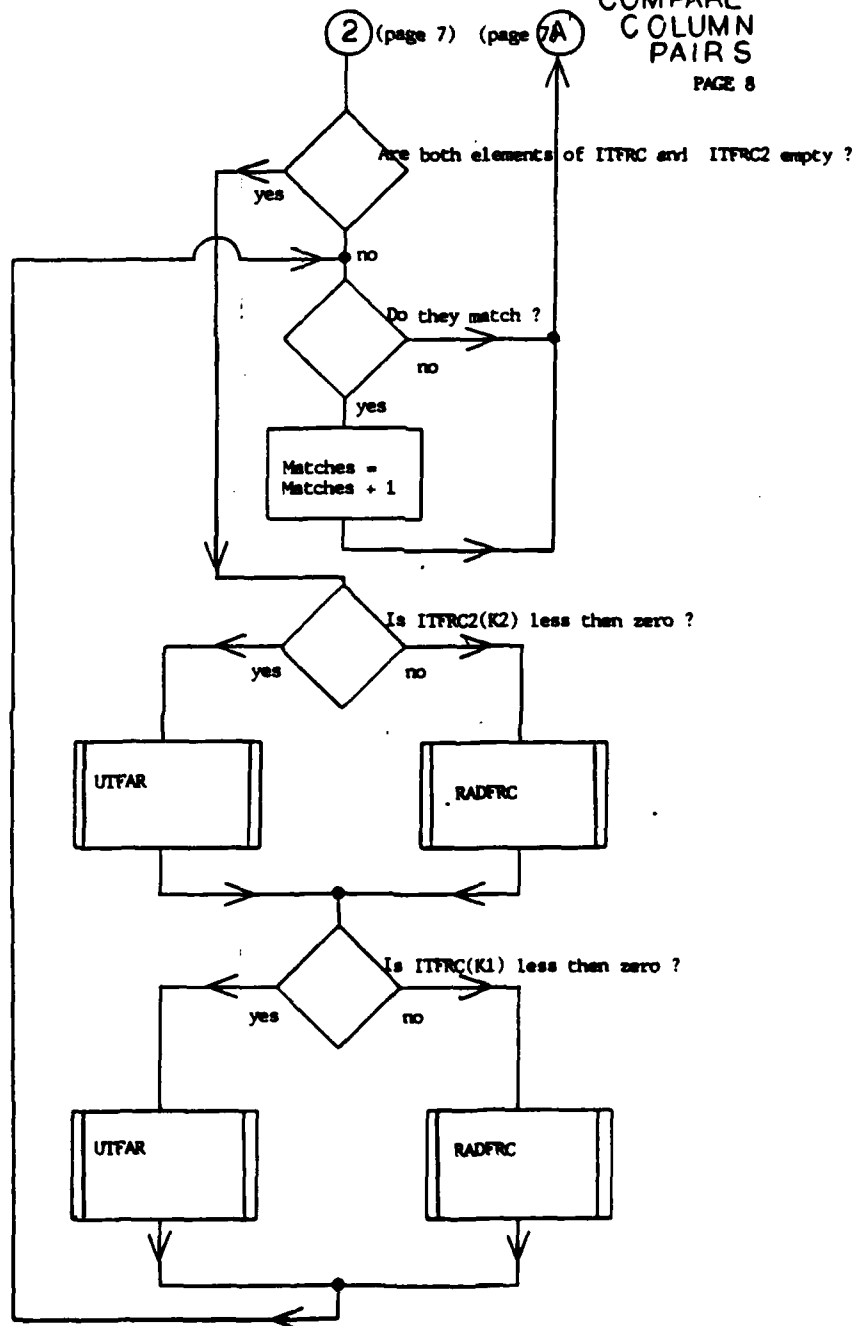




MERGEREL  
 COMPARE COLUMN  
 PAIRS  
 PAGE 7

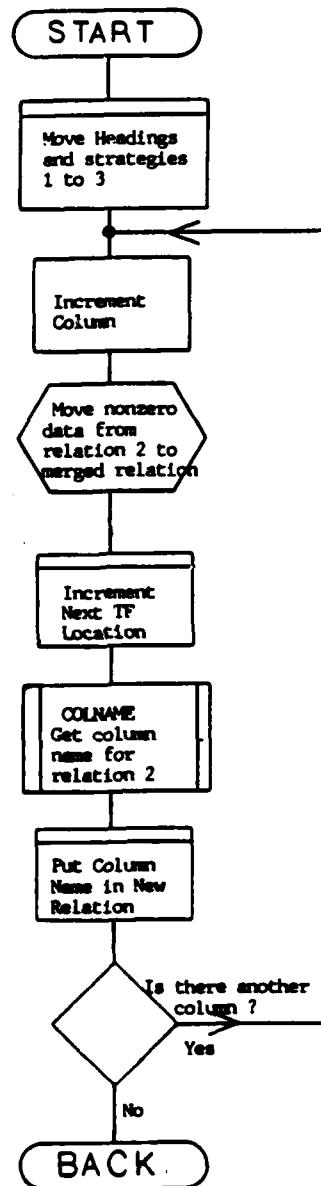


MERGEREL  
 COMPARE  
 COLUMN  
 PAIRS  
 PAGE 8



MERGEREL  
MOVE\_COLUMN.NAMES-  
AND-STRATEGIES

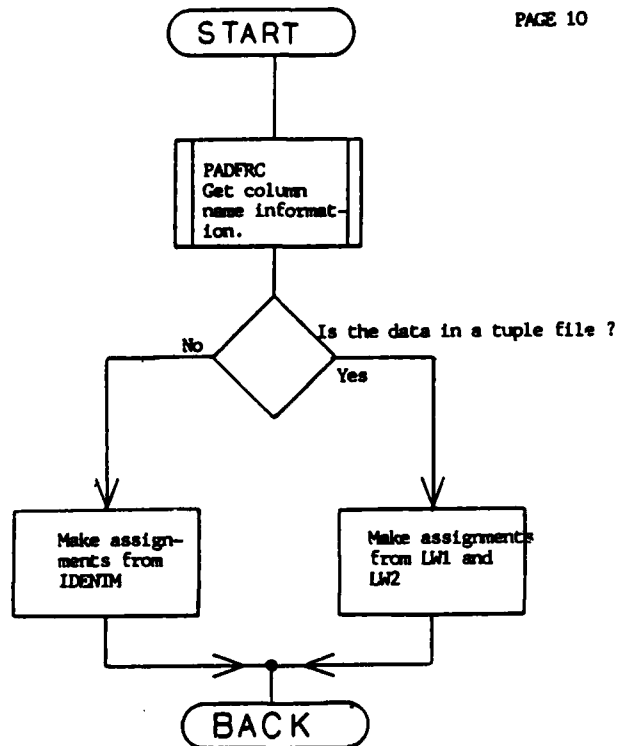
PAGE 9



# MERGEREL

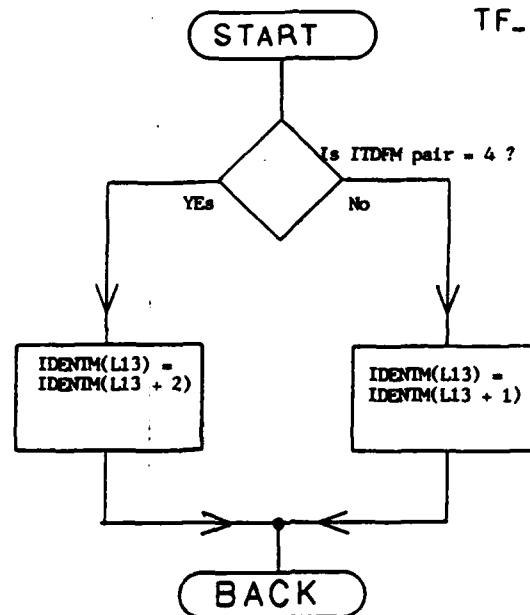
PUT\_COLUMN\_NAME  
IN\_NEW\_RELATION

PAGE 10

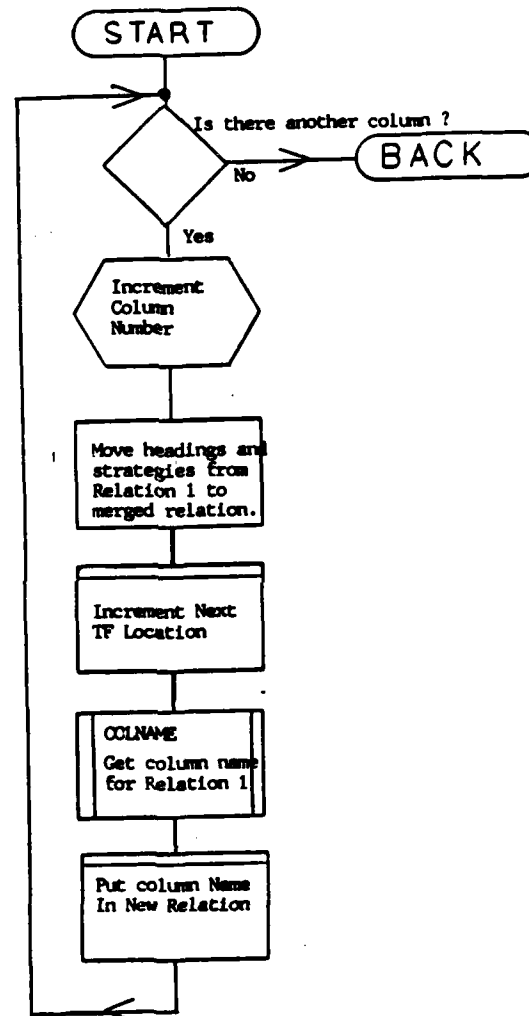


# MERGEREL

INCREMENT\_NEXT\_  
TF\_LOCATION  
PAGE 11



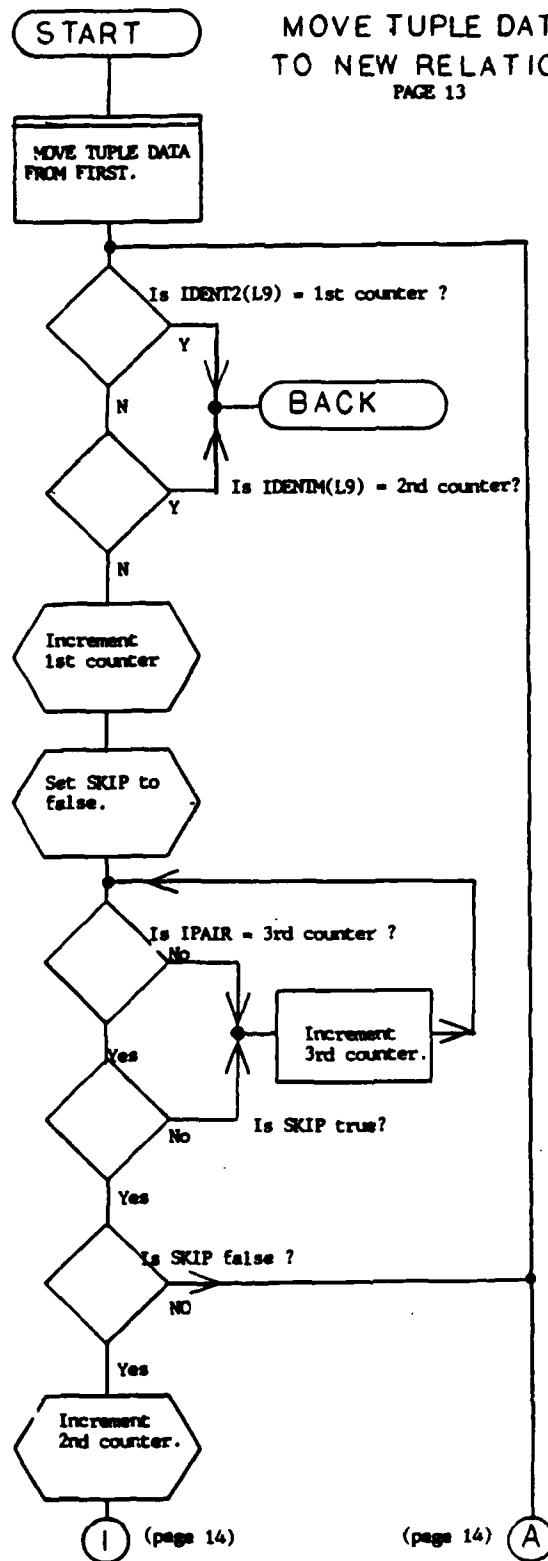
MERGEREL  
MOVE HEADINGS AND  
STRATEGIES 1 TO 3  
PAGE 12



# MERGEREL

## MOVE TUPLE DATA TO NEW RELATION

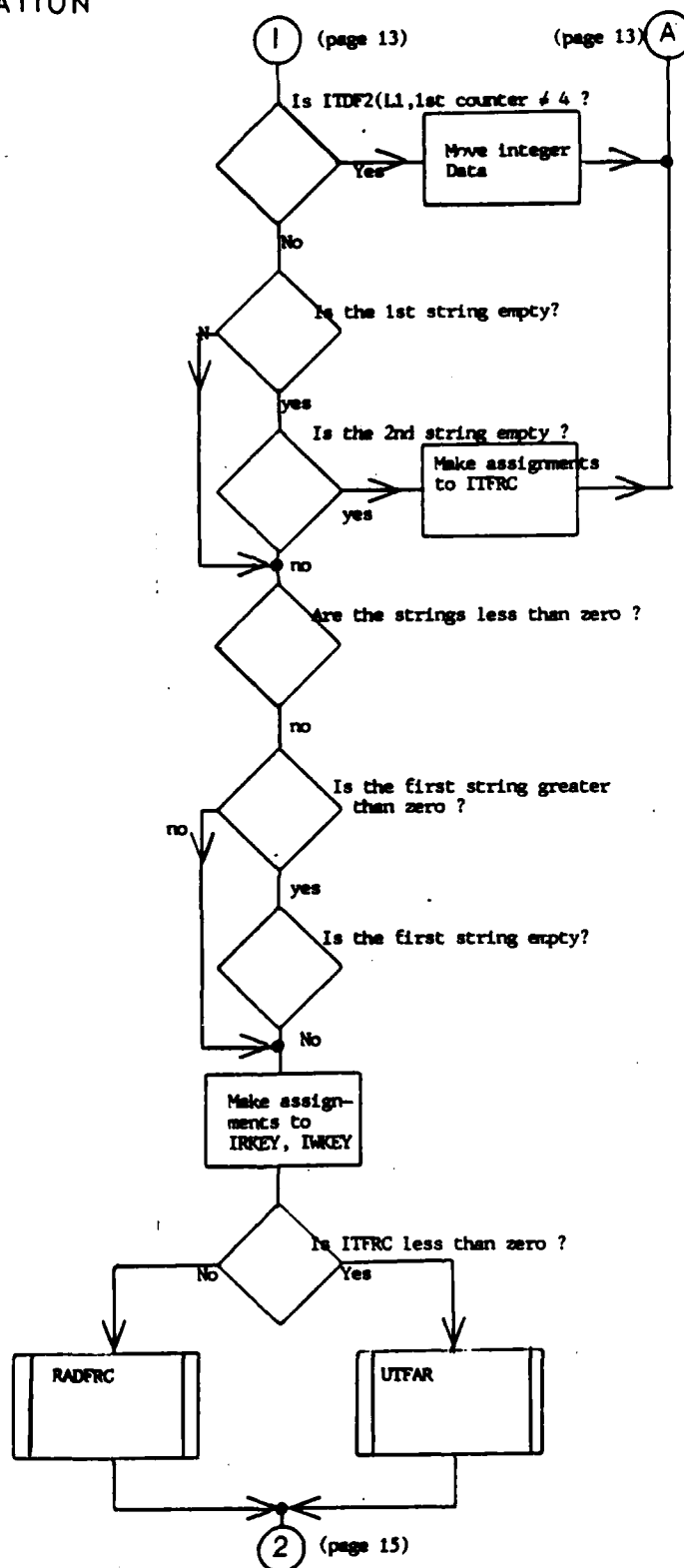
PAGE 13



MOVE TUPLE DATA  
TO NEW RELATION

MERGEREL

PAGE 14

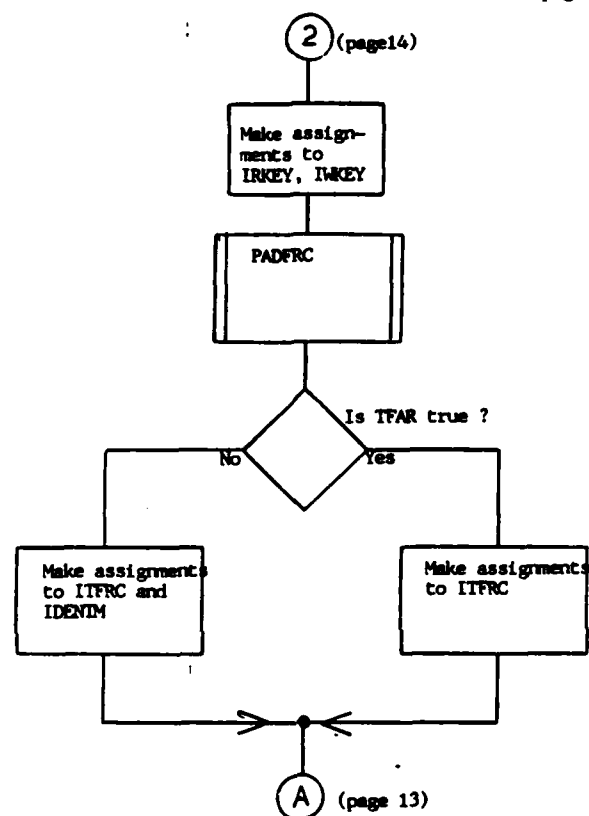




MOVE TUPLE DATA  
TO NEW RELATION

## MERGEREL

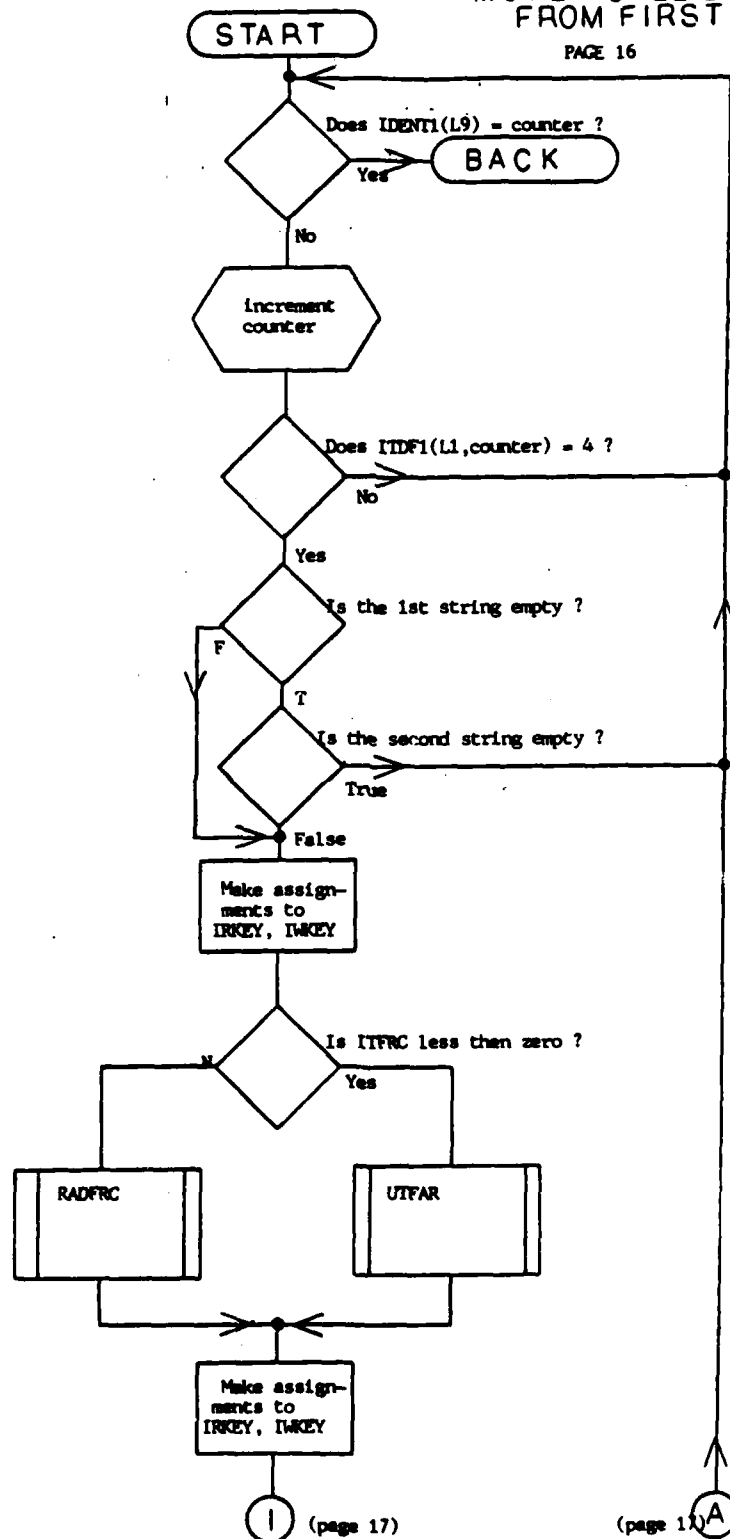
(page 15)



# MERGEREL

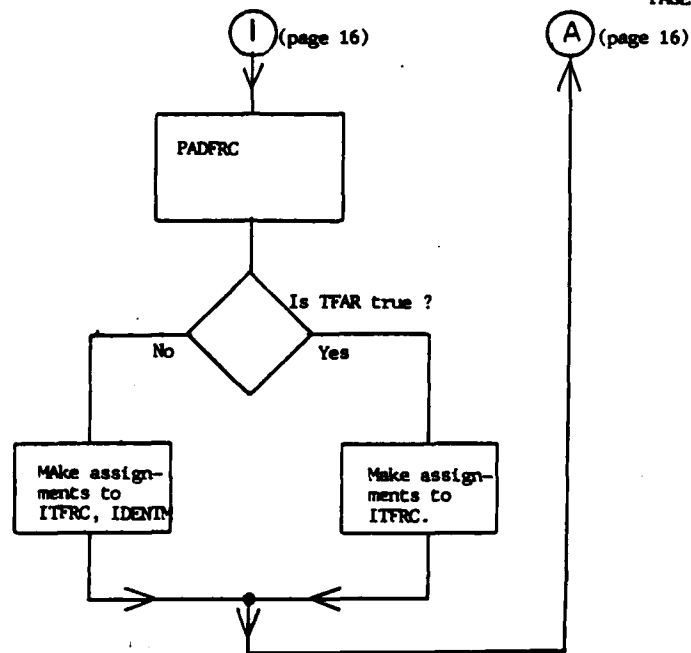
## MOVE TUPLE DATA FROM FIRST

PAGE 16



MERGE REL  
MOVE TUPLE DATA  
FROM FIRST

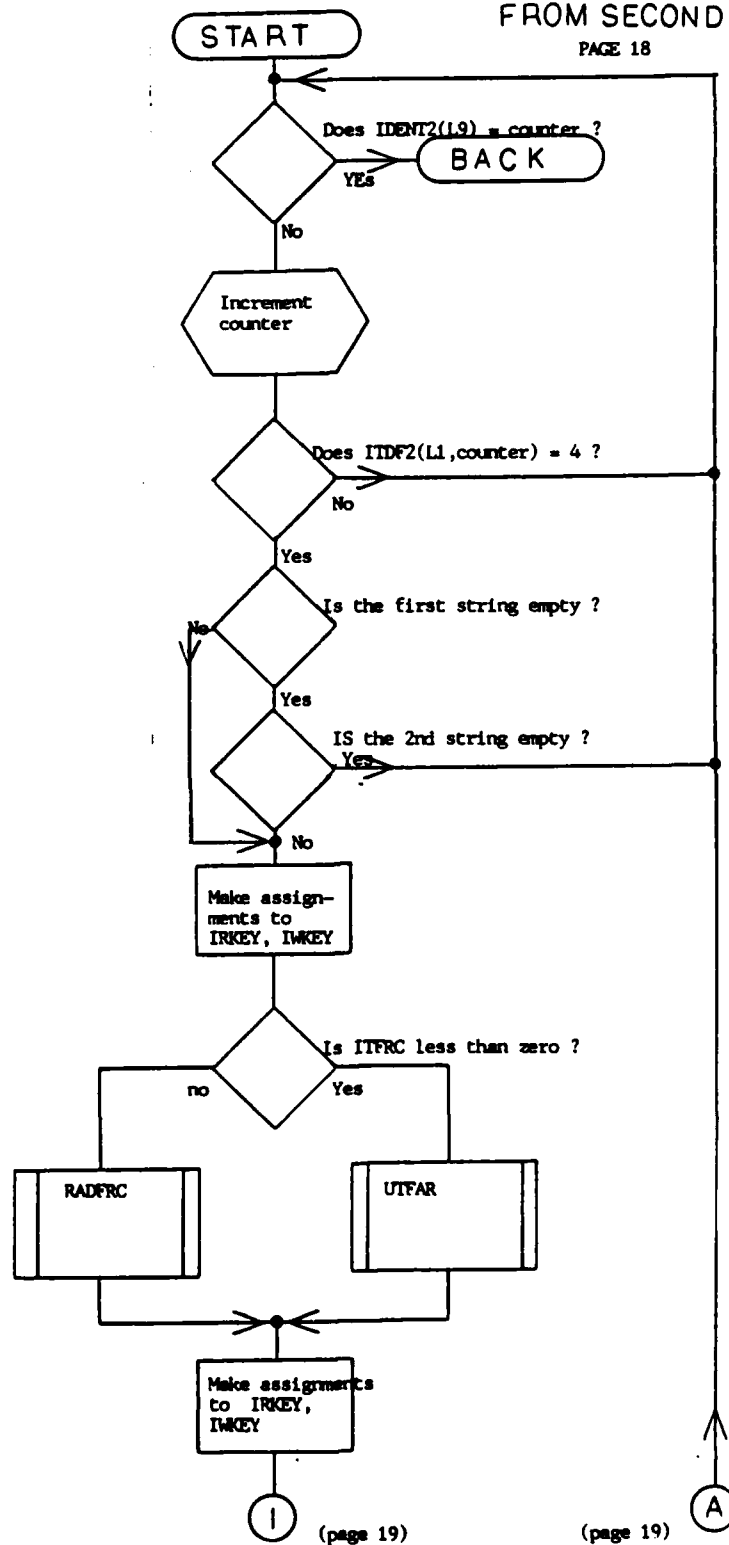
PAGE 17



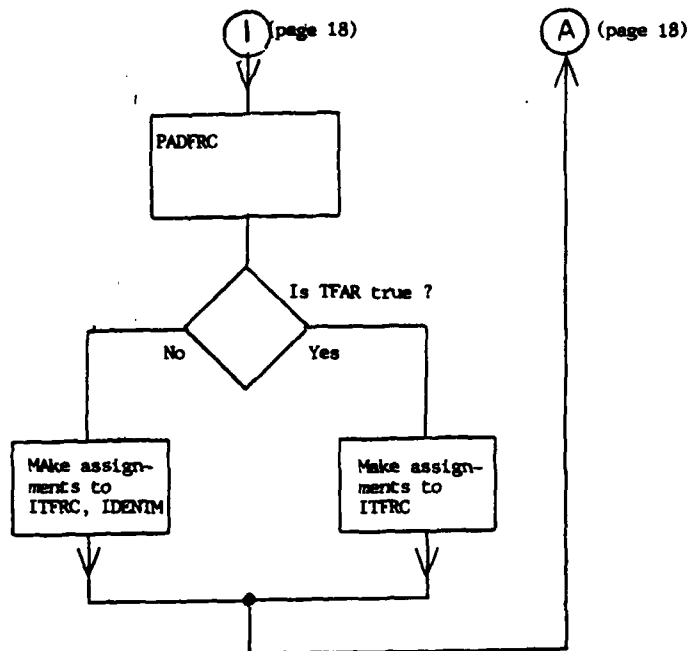
# MERGEREL

## MOVE TUPLE DATA FROM SECOND

PAGE 18



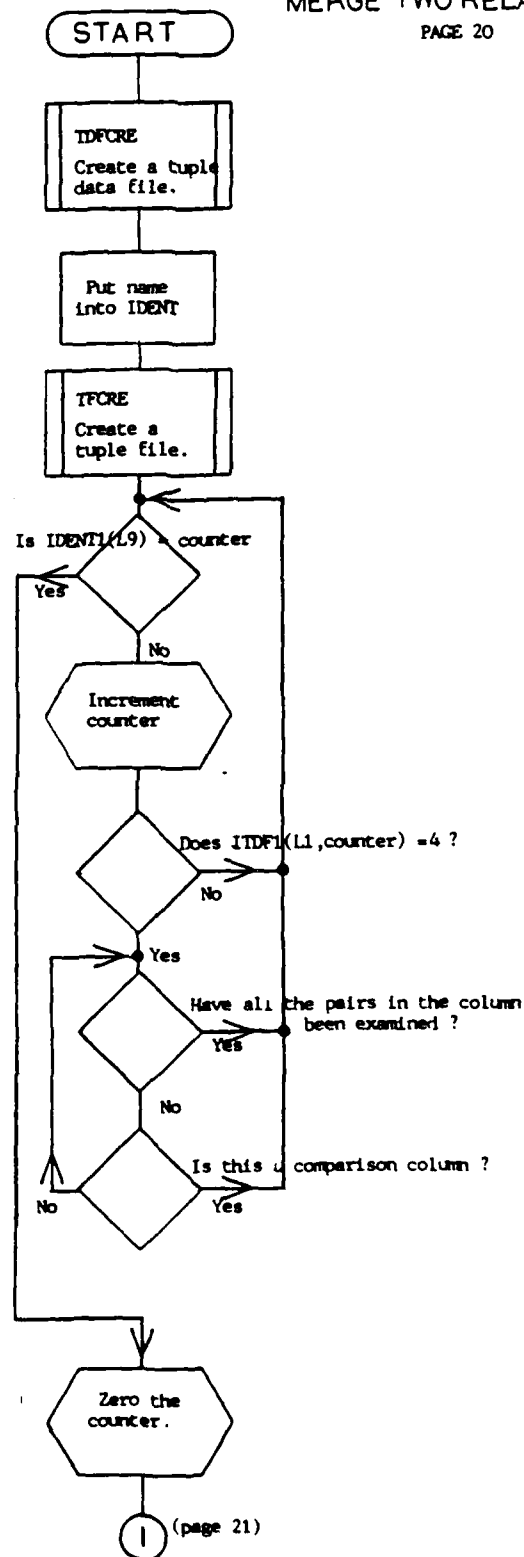
MERGEREL  
MOVE TUPLE DATA  
FROM SECOND  
PAGE 19



# MERGEREL

## MERGE TWO RELATIONS

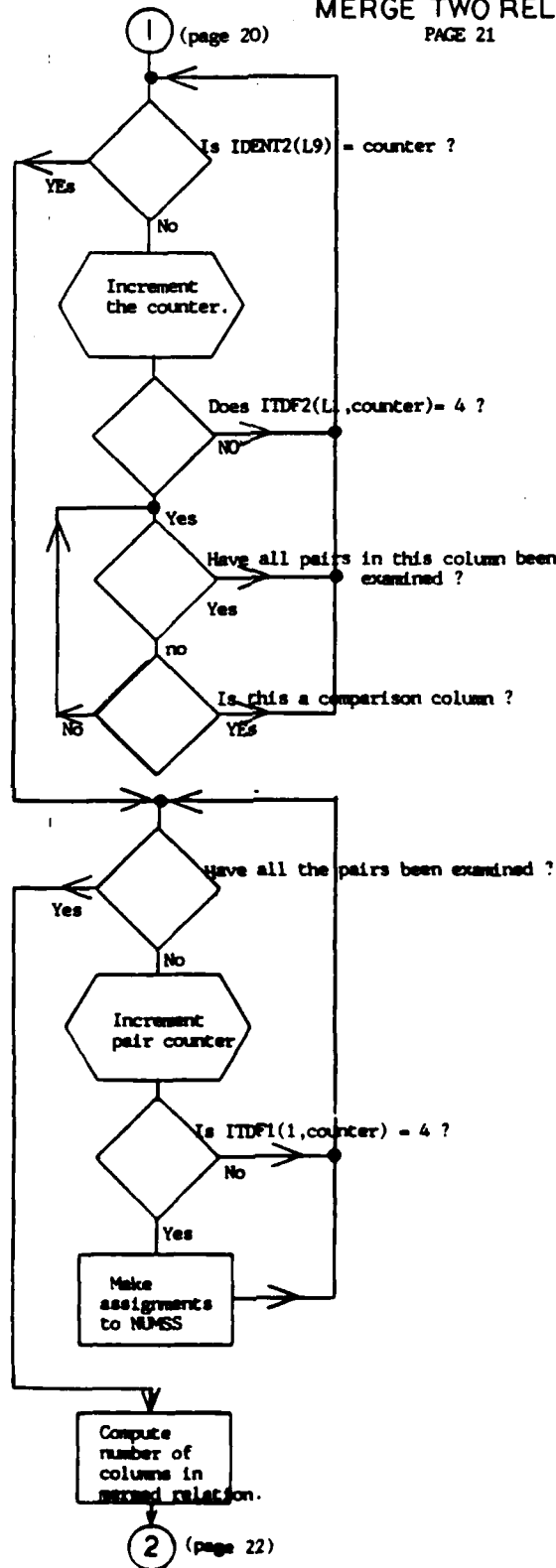
PAGE 20



# MERGEREL

## MERGE TWO RELATIONS

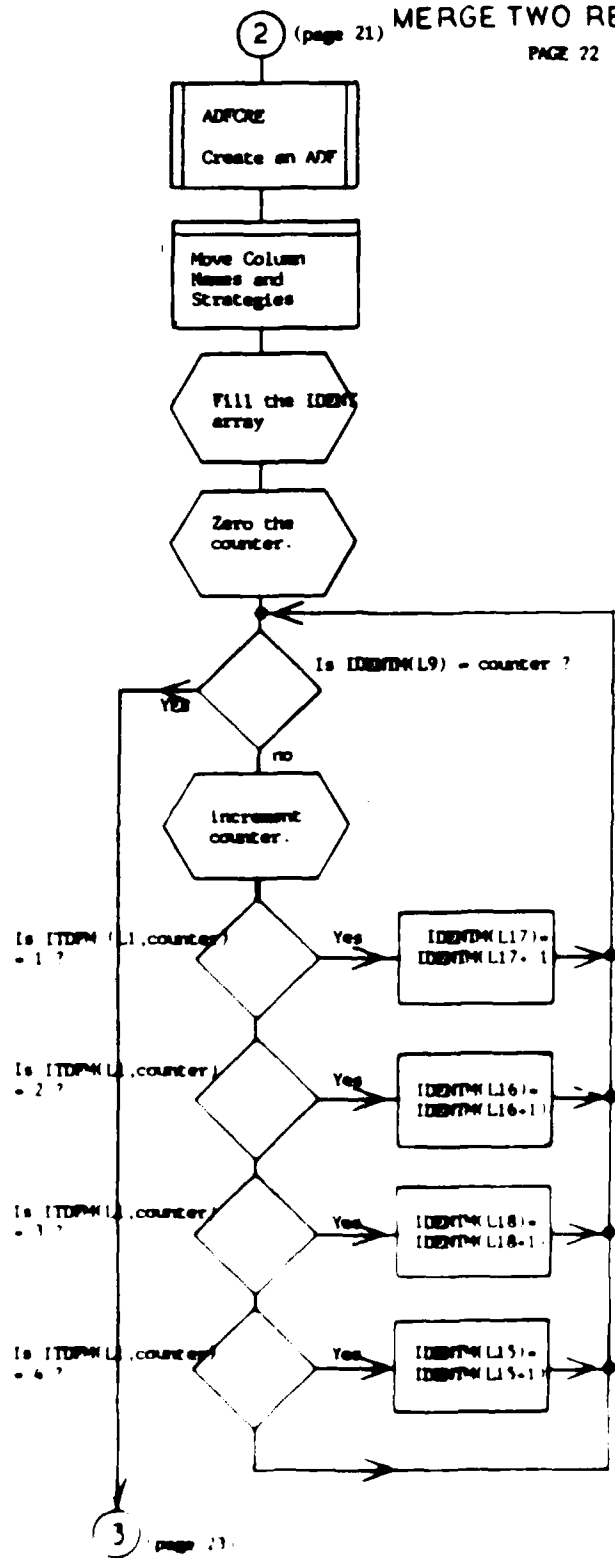
PAGE 21



# MERGEREL

## MERGE TWO RELATIONS

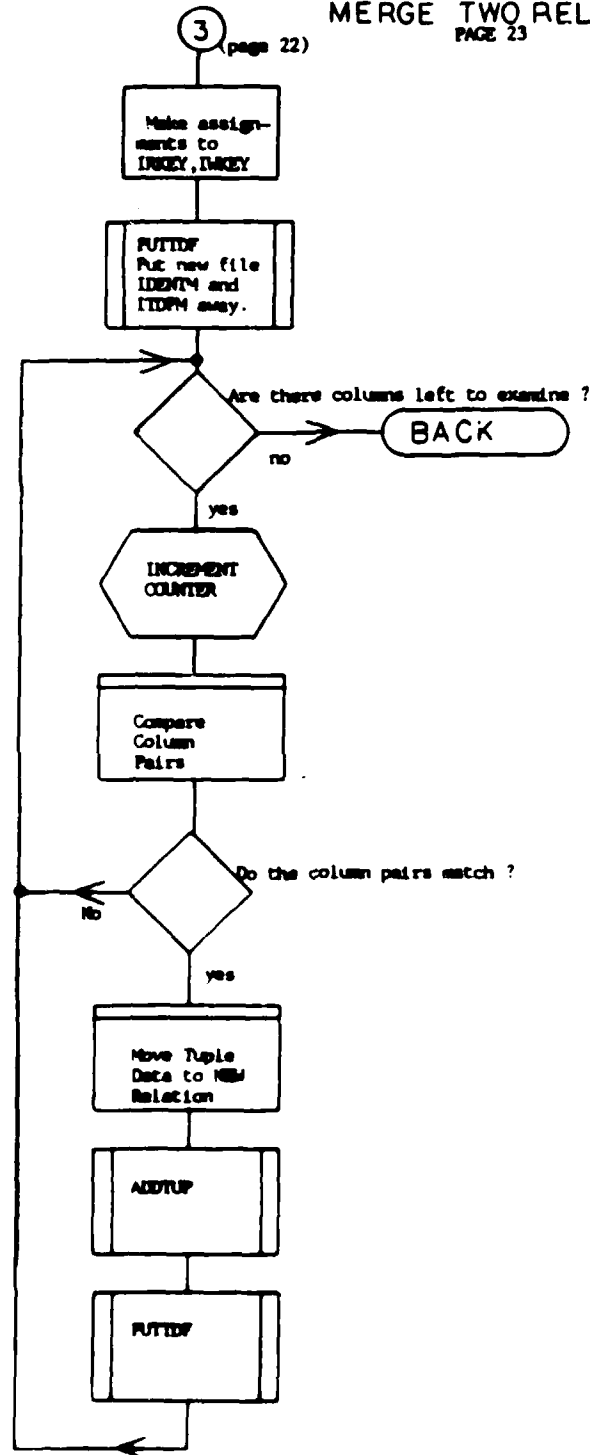
PAGE 22





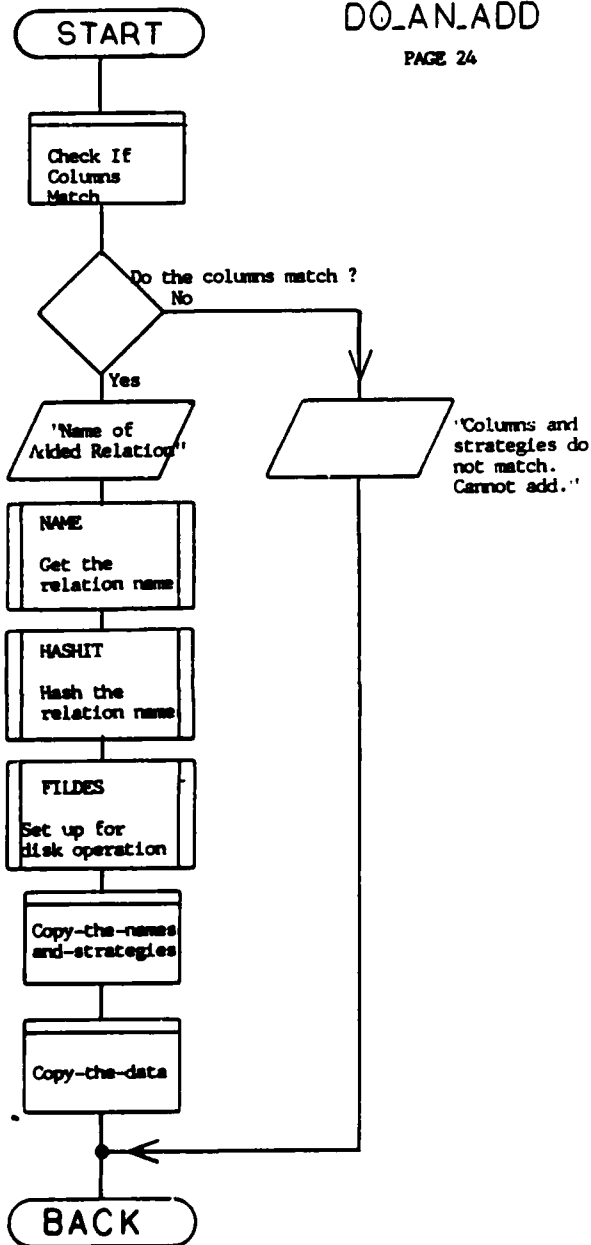
# MERGEREL

MERGE TWO RELATIONS  
PAGE 23



MERGEREL  
DO\_AN.ADD

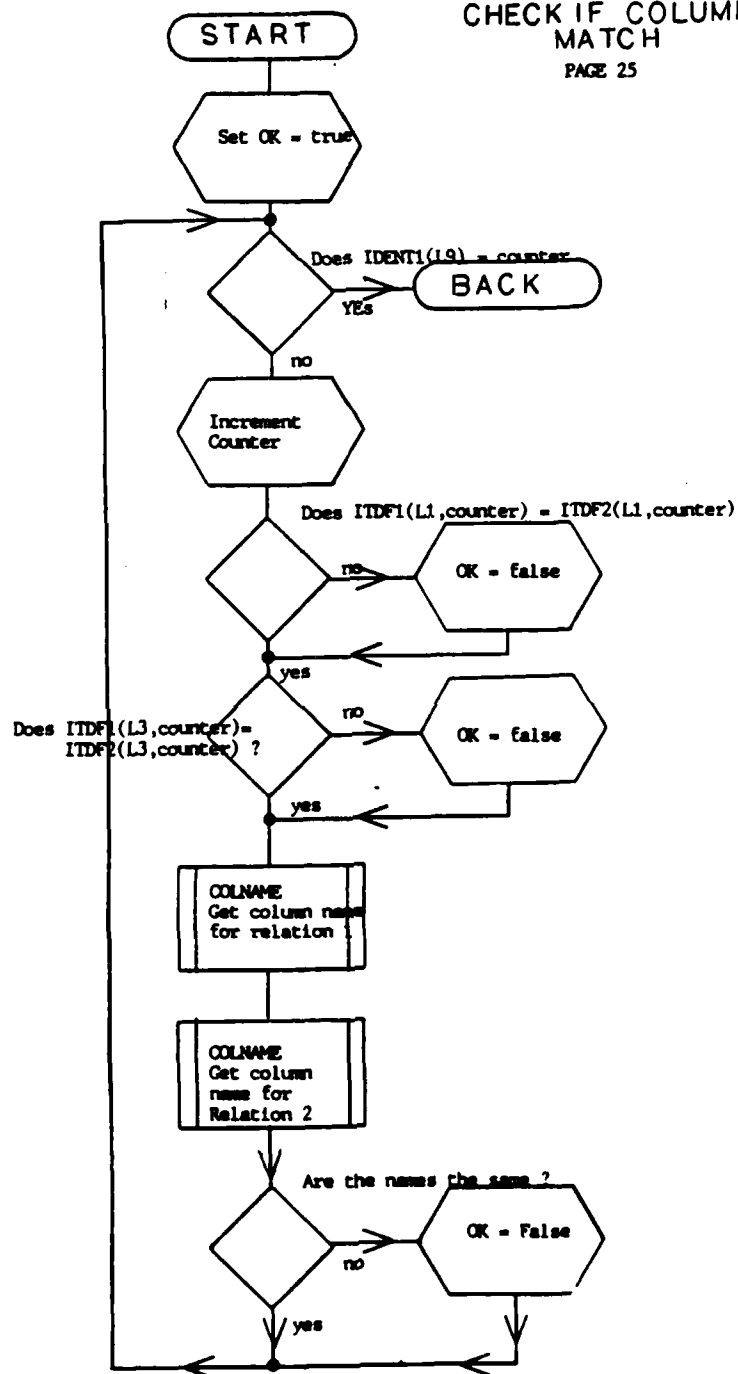
PAGE 24



# MERGEREL

## CHECK IF COLUMNS MATCH

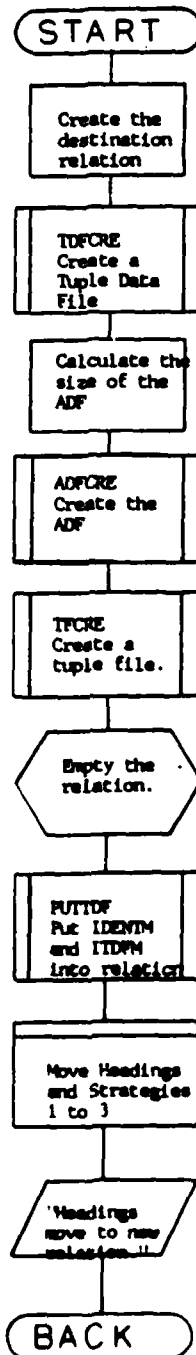
PAGE 25



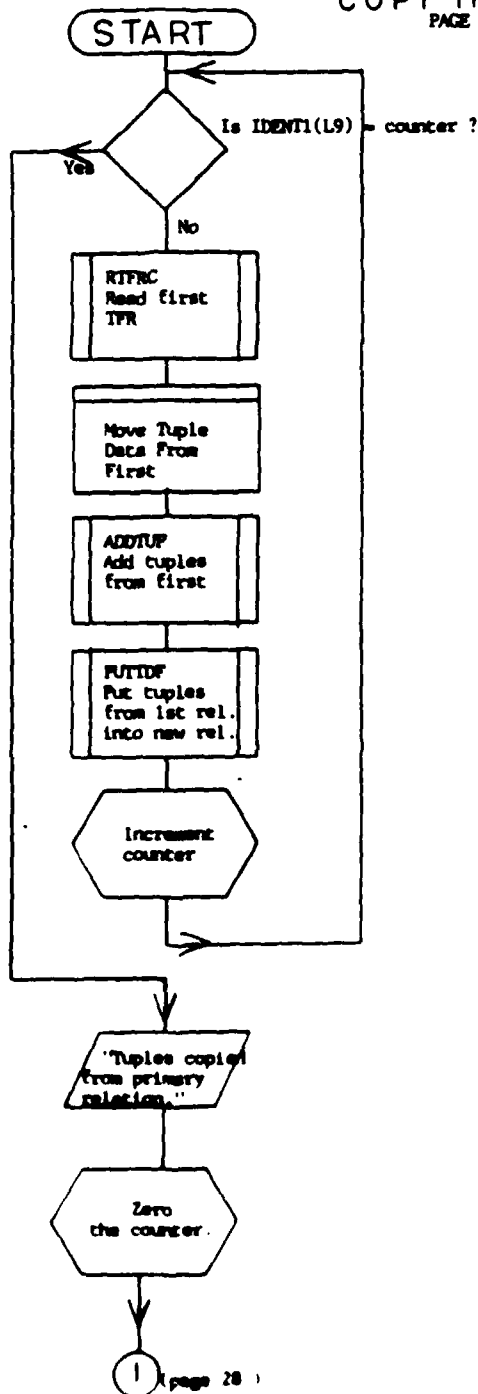
## MERGEREL

COPY THE NAMES  
AND STRATEGIES

PAGE 26

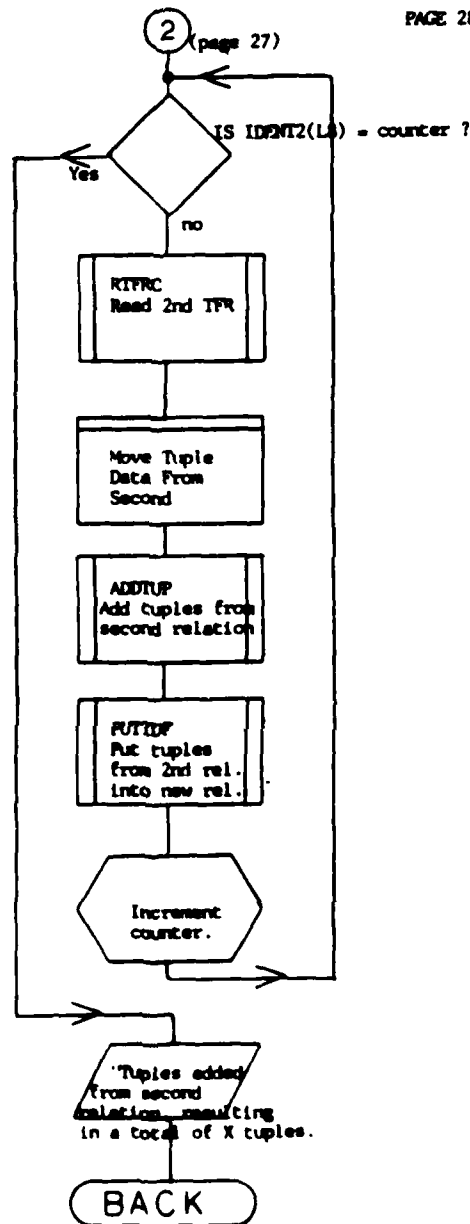


MERGEREL  
COPY THE DATA  
PAGE 27



MERGEREL  
COPY THE DATA

PAGE 28



**APPENDIX L**

**RORGRL**

Subroutine: REORGREL

This subroutine is used to reorganize a relation. The purpose is to remove unwanted "dead space" from the .ADF. The deleted records are removed from the .ADF file then the revised .ADF is written back for storage.



# REORG-11

Page 1

(START)

"Name of  
relation to  
be reorganized."

NAME  
Get the  
relation  
name.

HASHIT  
Hash the  
relation name.

FILDES  
Set up for  
disk operation

RIVTDF  
Bring tuple  
data into  
core memory.

Is there an I/O Error ?

Yes

RETURN

no

"Beginning  
reorganization."

EMLARY  
Empty the IDENT  
and new tuple  
descriptor array

Fill the ITDFI  
array with  
zeros.

1 (page 2)

Equate things  
in TDF that  
do not change

Transfer the  
relation name  
into IDENT array.

RESET TEMPFIL  
Clear the  
temporary file.

Open the tuple file, change access, prepare RELDESA for rename	OPNTF MODAP CATSTR FSLIDE
---	------------------------------------

DELETE TEMPFIL  
IF IT EXISTS

Rename RELDESA TF to RELDEB. TF	FILEREN FILECLS
---------------------------------------	--------------------

RESET TEMPFIL  
Clear the  
temporary file.

Open tuple data file, change access prepare RELDES for rename.	OPNTDF MODAP CATSTR FSLIDE
---	-------------------------------------

DELETE TEMPFIL  
IF IT EXISTS.

Rename RELDESA TDF to RELDESB TDF	FILEREN FILECLS
---	--------------------

RESET TEMFILE  
Clear the  
temporary file.

Open the ADF,  
change access,  
prepare RELESA  
for rename.

OPNADF  
MODAP  
CATSTR  
FSLIDE

DELETE TEMFILE  
IF IT EXISTS

Rename RELESA  
ADF to RELESA  
ADF

FILEREN  
FILECLS

TFPRE  
Create a  
tuple file

TDPRE  
Creates the  
tuple data  
file

Calculate the  
size of the  
ADF.

ADFPRE  
Create the ADF

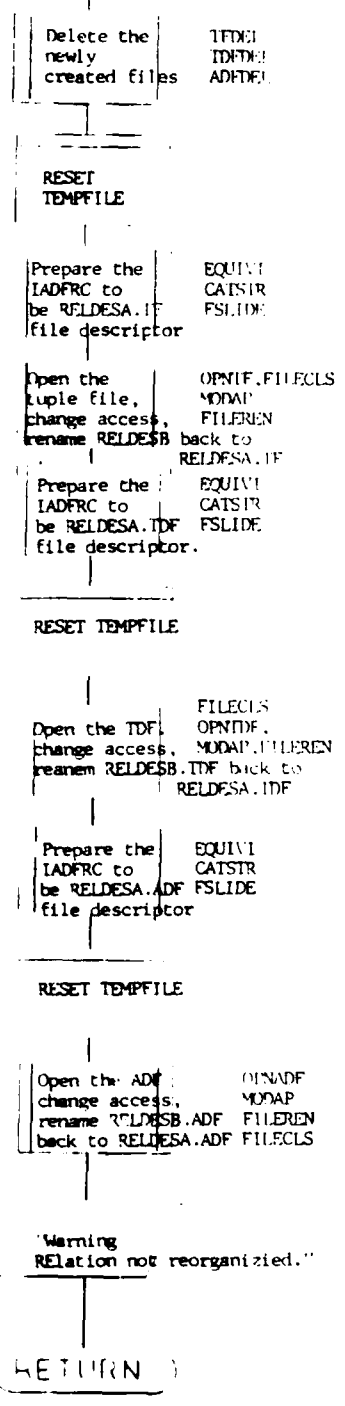
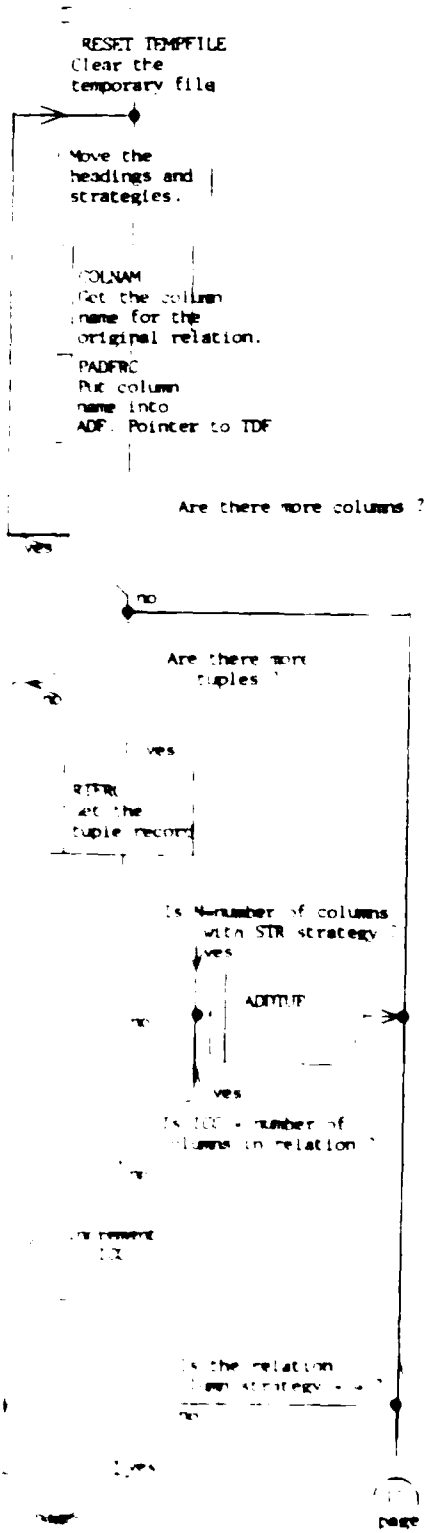
Were there any errors in TFPRE, TDPRE  
ADFPRE ?

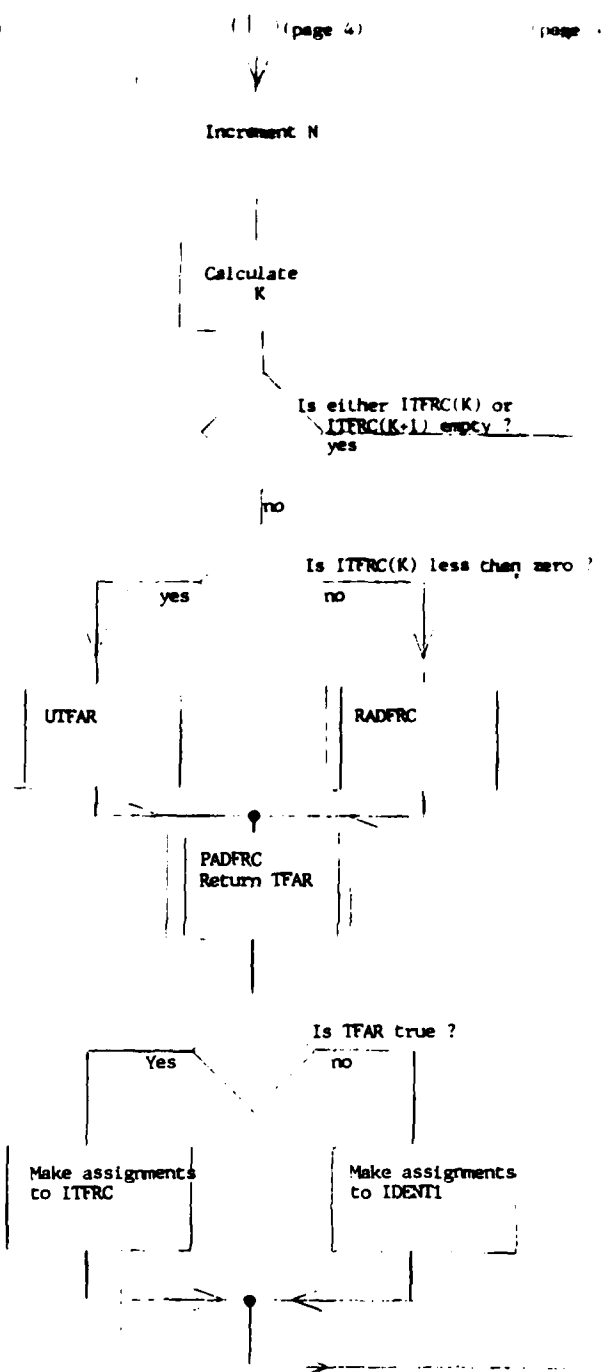
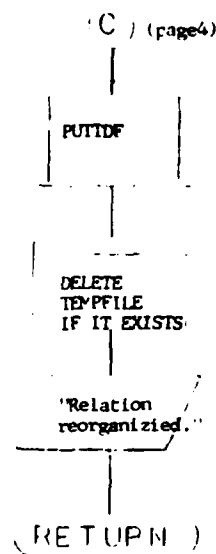
NO

YES

(page 4)

(page 4)





PAGE 1

DELADY  
Delete the  
tempfile

TPDEL  
Delete the  
temp file

ADTDEL  
Add to  
the string

ADTDEL  
Add to  
the string

BACK

START

RESET TEMPFILE

TPDEL  
Delete the  
temp file

TPDEL  
Delete the  
temp data file

ADTDEL  
Delete the ADT

BACK

START

FILED  
Delete the  
Temporary File

Set 1000000  
(There is no  
I/O Error)

BACK

APPENDIX M

BKUPRL



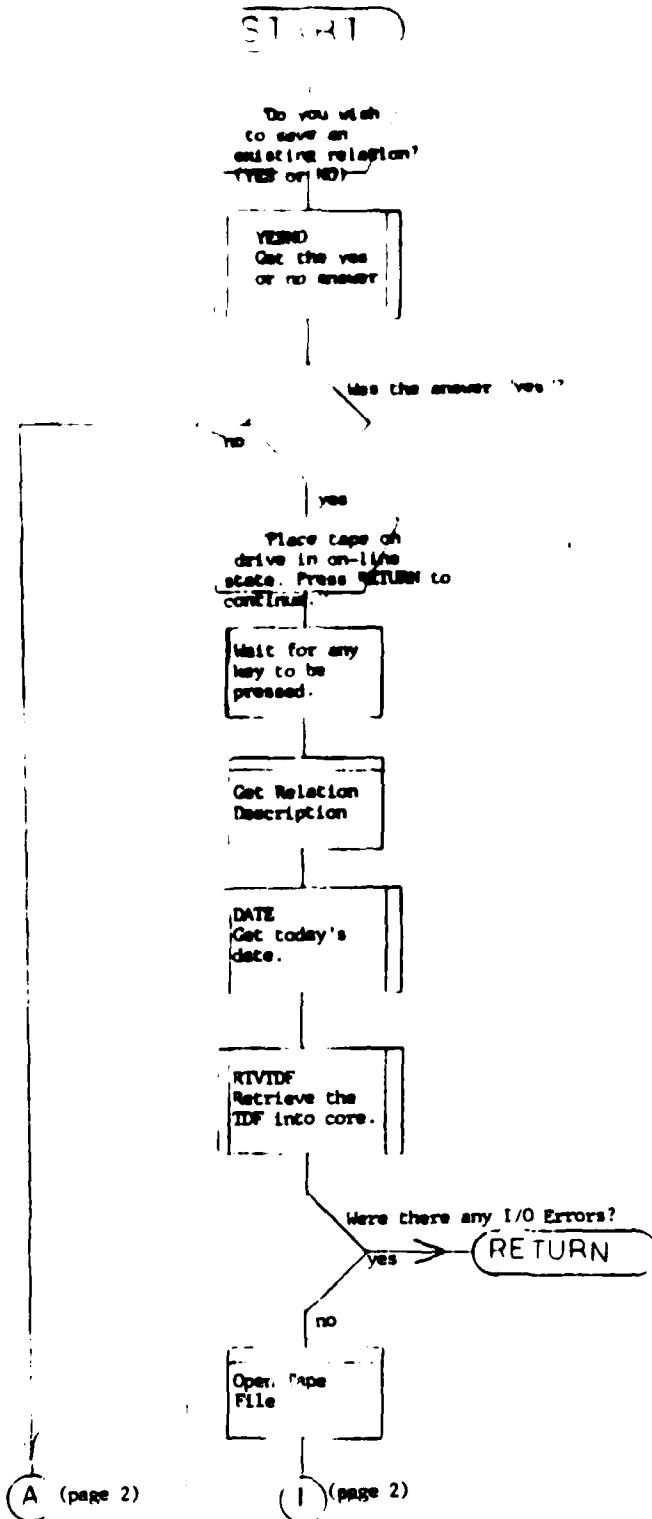
### Subroutine : BACKUPREL

This subroutine allows the user to backup or restore relational data. In backing up a user will copy the data from the primary storage media (usually system disk) to a secondary storage media (magnetic tape.) This protects the user's data and allows reconstruction in the event of a system failure.

In restoring a relation, data is transferred from magnetic tape back to the system. This utility is used primarily after a system failure. This subroutine is setup to transfer the .TF, .ADF, and .TDF files from the user storage volume to the nine track tape drive. The user must insure mounting of the proper tape and placing the drive in the proper configuration.

The following information is stored on magnetic tape during a backup:

1. The relation name
2. The date of backup
3. Read/write security keys
4. The Tuple Descriptor File (TDF)
5. The Tuple File (TF)
6. The Alpha Data File (ADF)



A (page 1)

(1) (page 1)

Set read and  
write heads  
to start  
to tape

Write out the  
relation name  
and currently  
have to tape

RTVTD  
Get the TDF  
into core

Write the TDF  
out to tape

Have all the tuples  
been written out to  
tape?

yes

no

RTFRC  
Get the next  
tuple into  
core

Write the  
tuple out  
to tape

Have all the  
ADF elements  
been written out  
to tape?

yes

no

RADRC  
Get the ADF into  
core

(page 3)

A

(page 3) C

(page 3) 2

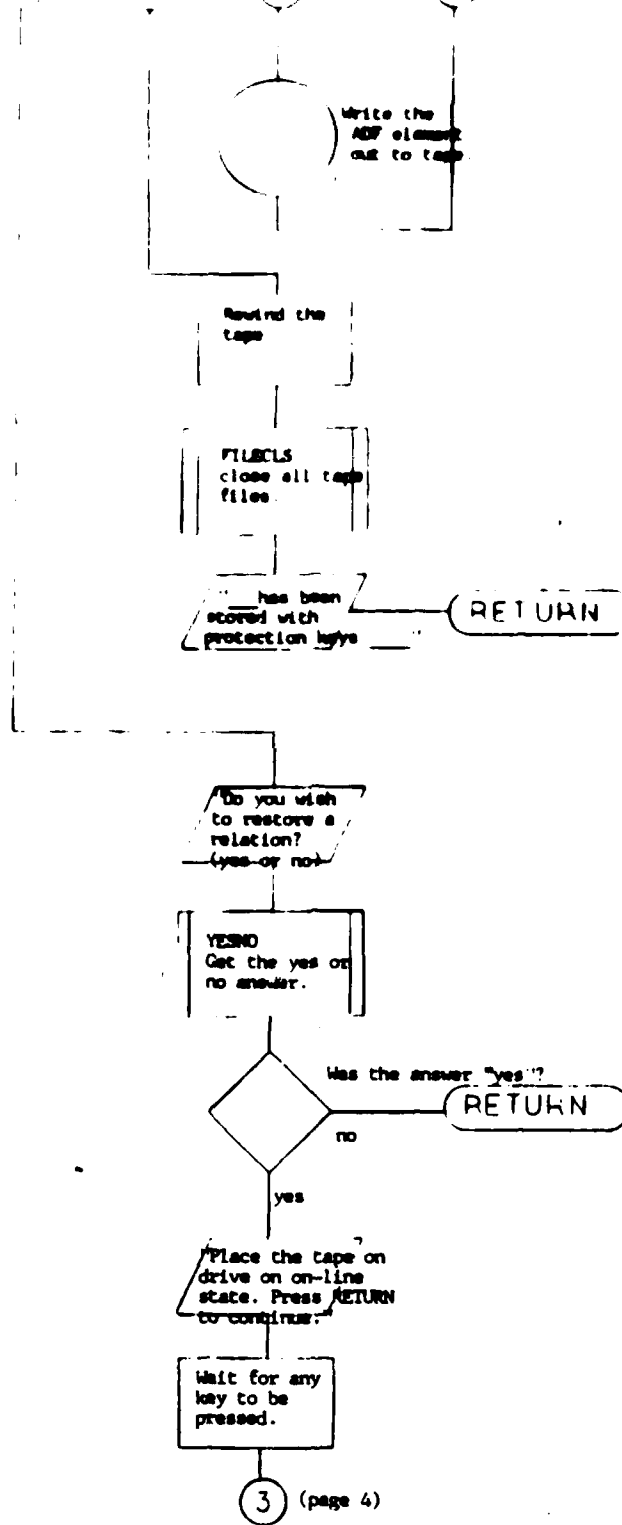
B (page 3)

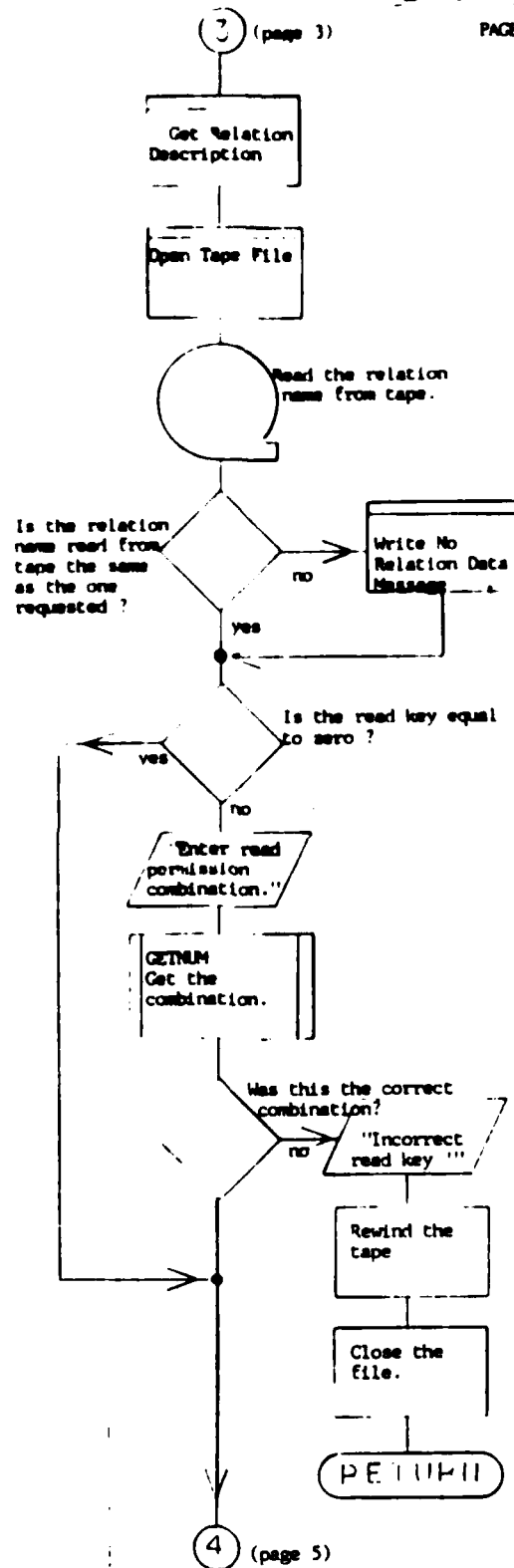
(A)

(C)

(2)

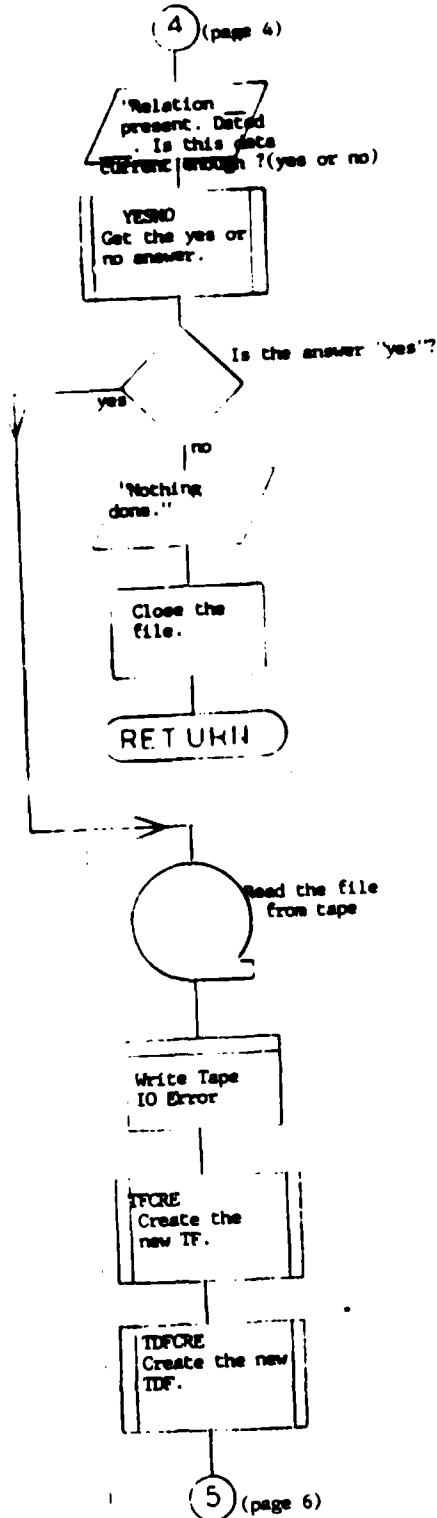
(B)

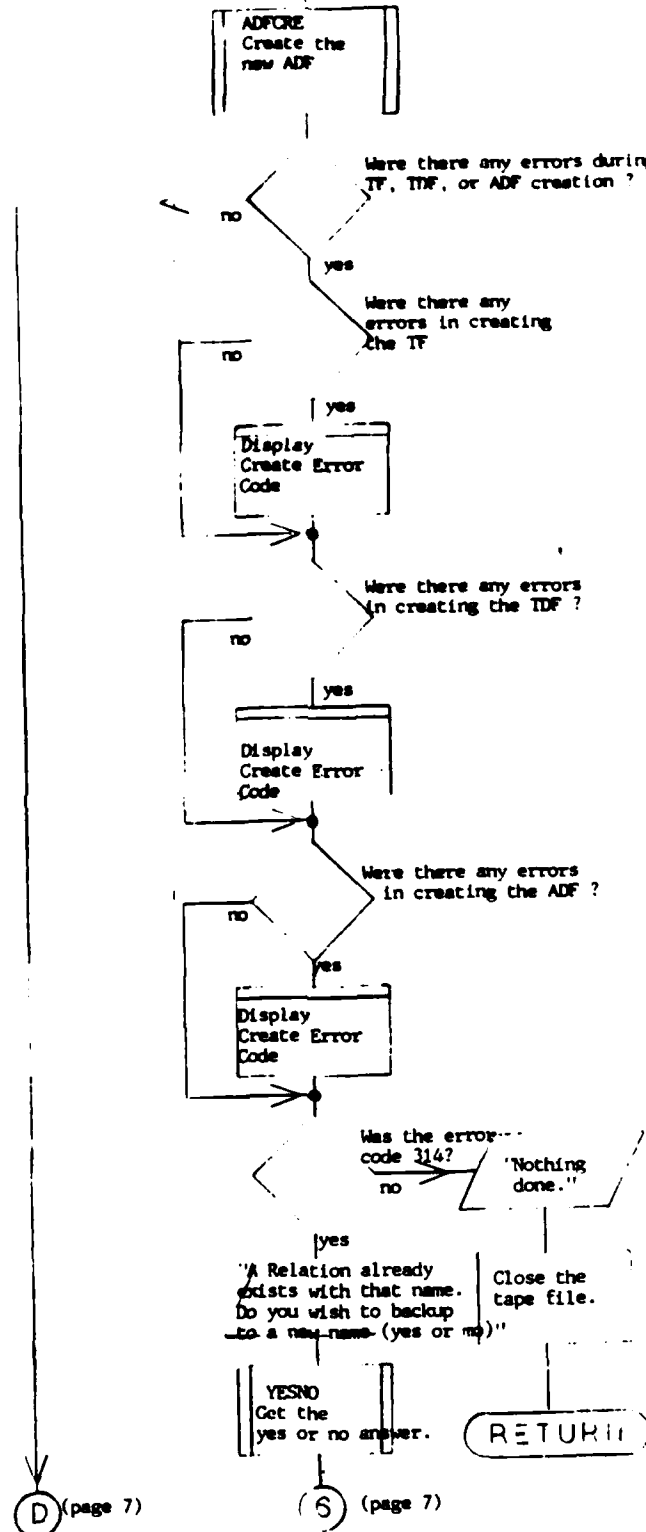


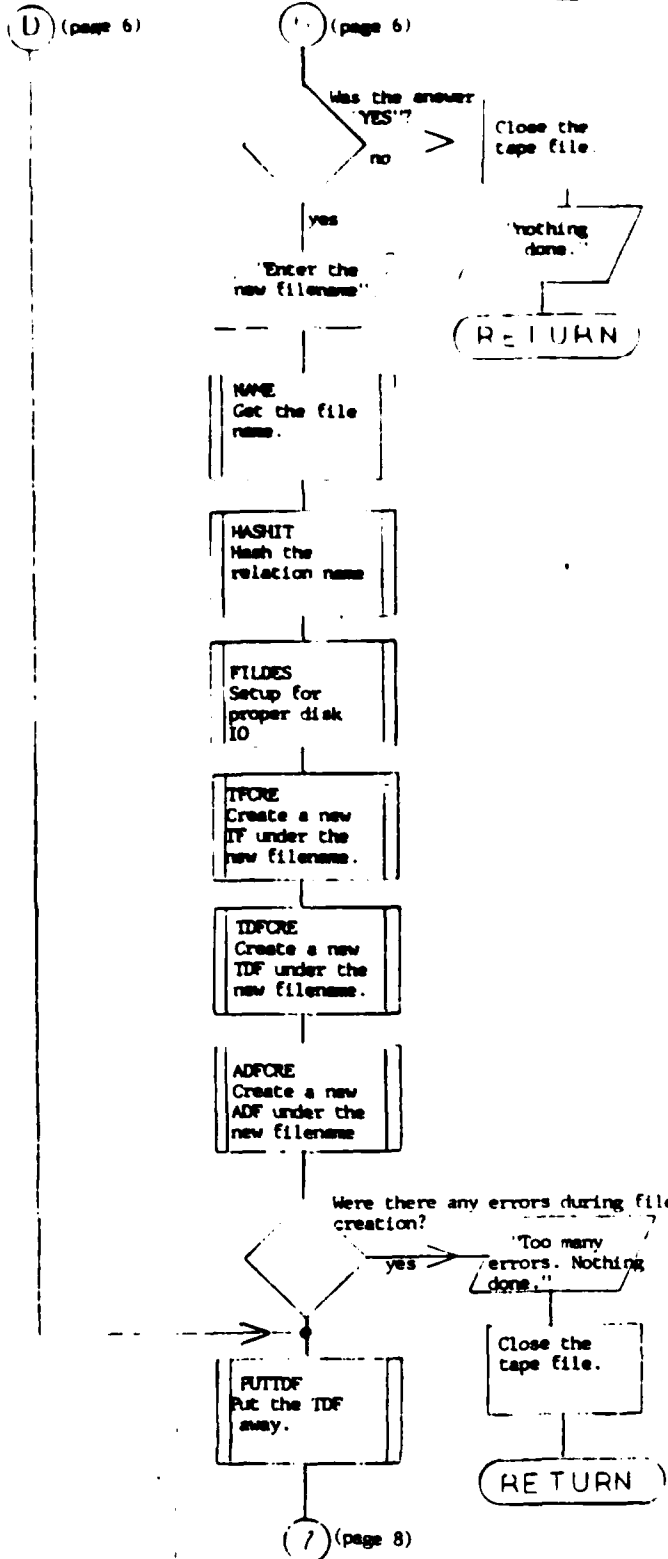


# BACKUP.FL

PAGE 5









page 1

page 2

Find the value  
of each reading  
of records

Write Tape  
10 Error

Have all the  
elements been  
read in?

Read in another  
ADF element

Put the 10  
element into  
relation

Read in another  
ADF element

Write Tape  
10 Error

Have all the ADF  
elements been  
read in?

Read in another ADF  
element

(page 9) E

(page 9) B

F (page 9)

# BACKUP

PAGE 9

(page 8) E (page 8) E (page 8) F

PADPNC  
Put away the  
ADF element.

Rewind  
the tape

FILECLS  
Close all  
tape files.

Relation  
has been stored  
with data, dated ."

RETURN

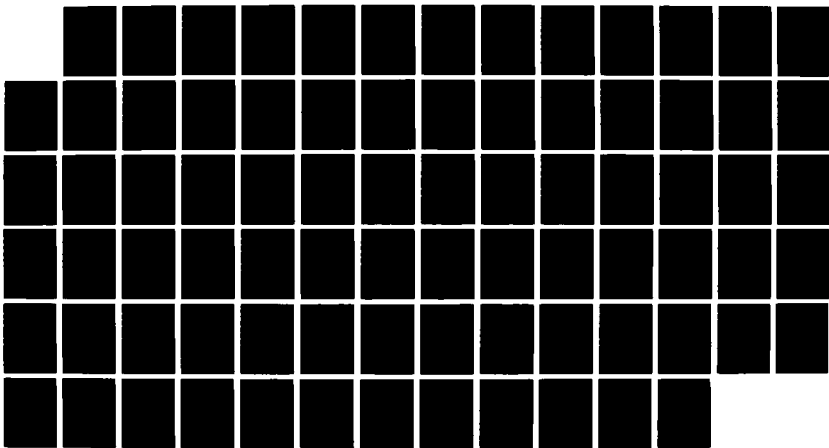
AD-A188 002

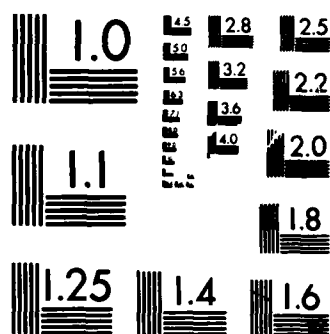
FEASIL IMPLEMENTATION UNDER VAX VMS WITH DESIGN  
INFORMATION(U) ALABAMA UNIV IN HUNTSVILLE DEPT OF  
ELECTRICAL AND COMPUTER EN. J D MARR ET AL. NOV 86  
UAH-5-31325 AMSMI-CR-RD-55-86-5 F/G 12/5

4/4

UNCLASSIFIED

NN





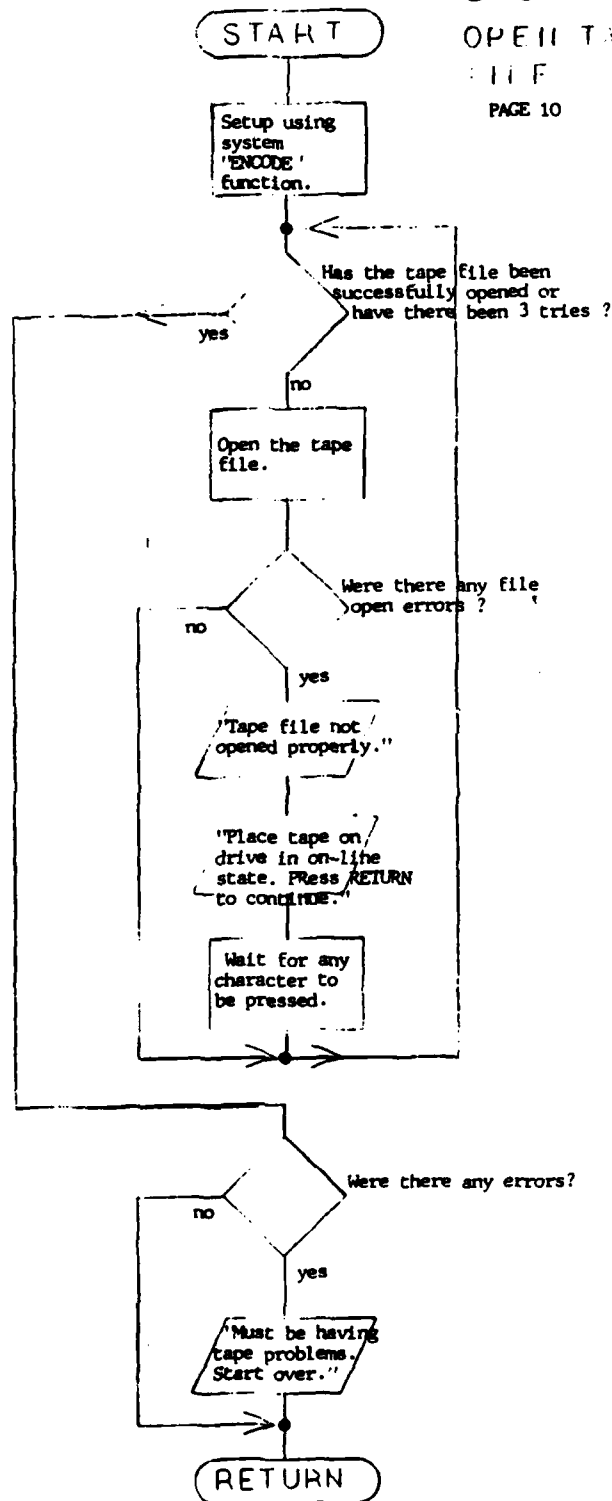
MICROCOPY RESOLUTION TEST CHART  
NATIONAL BUREAU OF STANDARDS-1963-A

BACKUPRII

OPEN TAPI

FILE

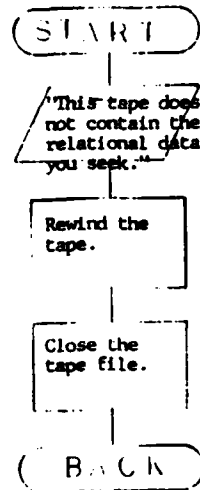
PAGE 10



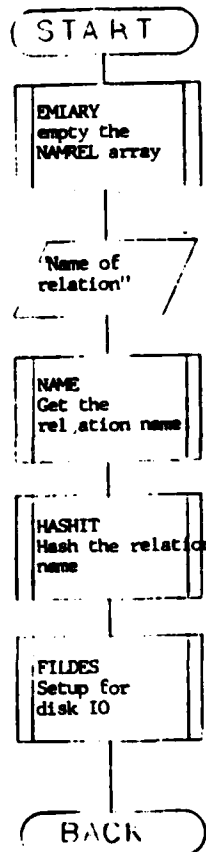
# BACKUP FILE

WRITE II  
RELATION  
DATA  
MEMO A

PAGE 11

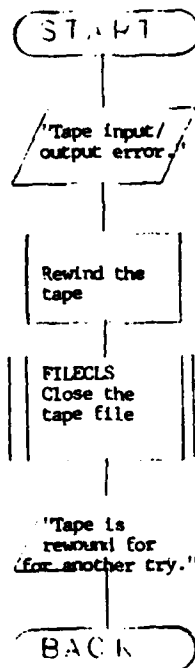


GET  
RELATION  
DESCRIPTION

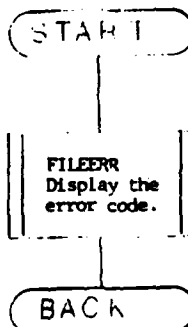


BACKUP REI  
WRITE TAPE  
IN ERROR

PAGE 12



DISPLAY  
CREATE  
ERROR  
CODE



**APPENDIX N**

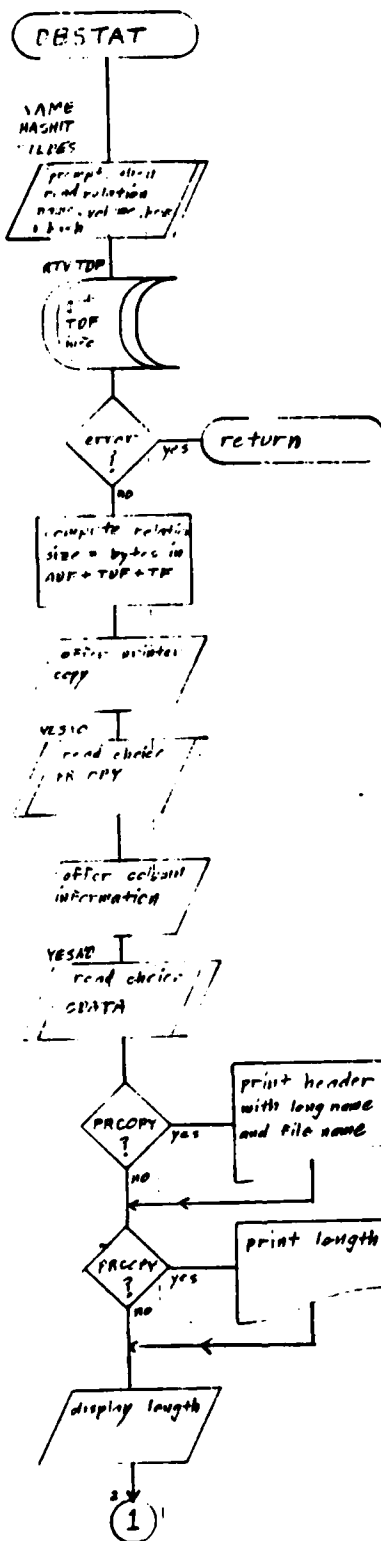
**DBSTAT**



### Subroutine: DBSTAT

This routine gives the user the physical status of a relation. This information includes the size of the relation, the number of columns, the number of rows, the dead to active space ratio, as well as the column data, if desired.

DBSTAT allows for a printer copy of the relation status, if desired.



DBSTAT p.1

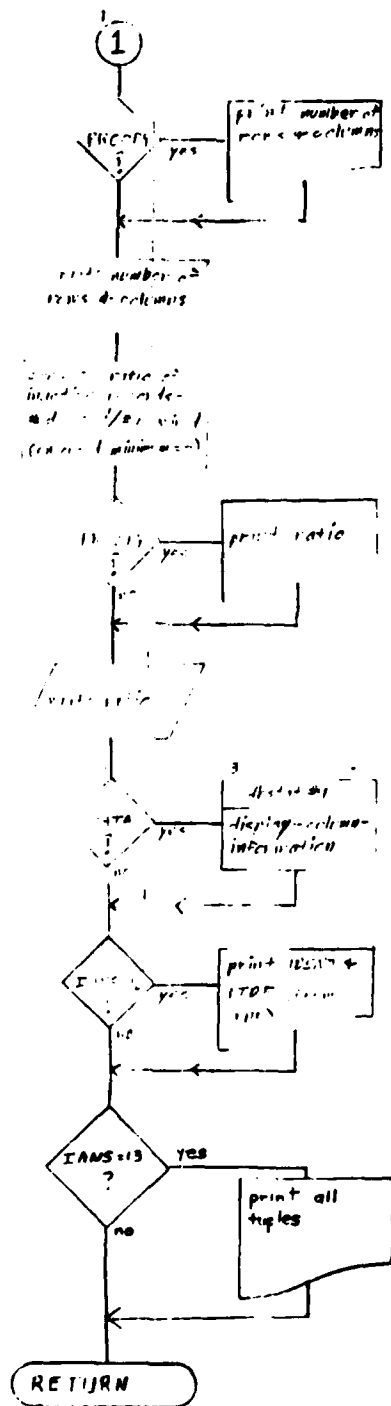
File: DBACNT:11

Gives relation physical details.

Notes: 2 return points.

Significant inputs:

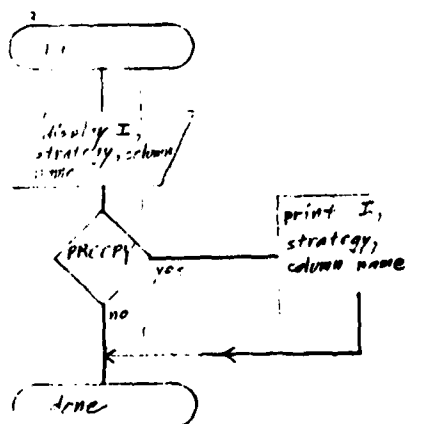
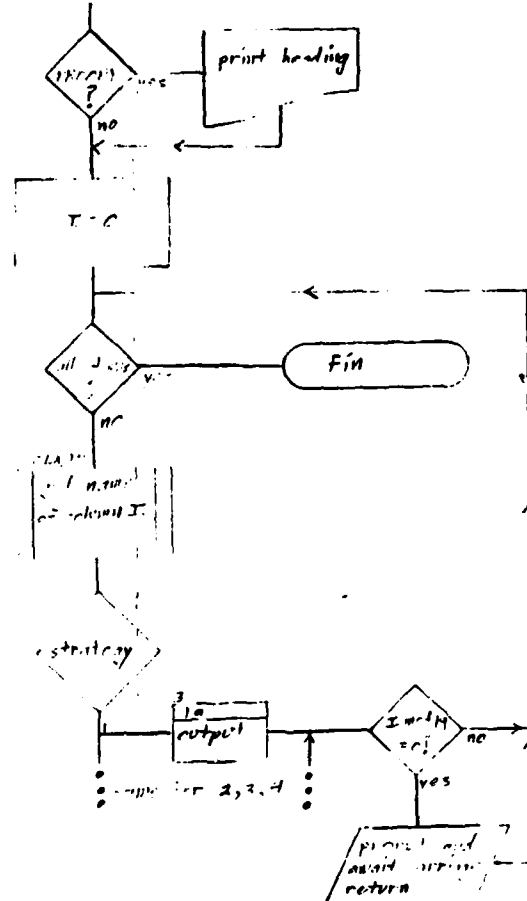
WAYS (11 or 12)  
P or S.



Module #1

DESTAT p.3

#1- display-column-information



APPENDIX O

HELP

OVERLAY: HELP.FLC

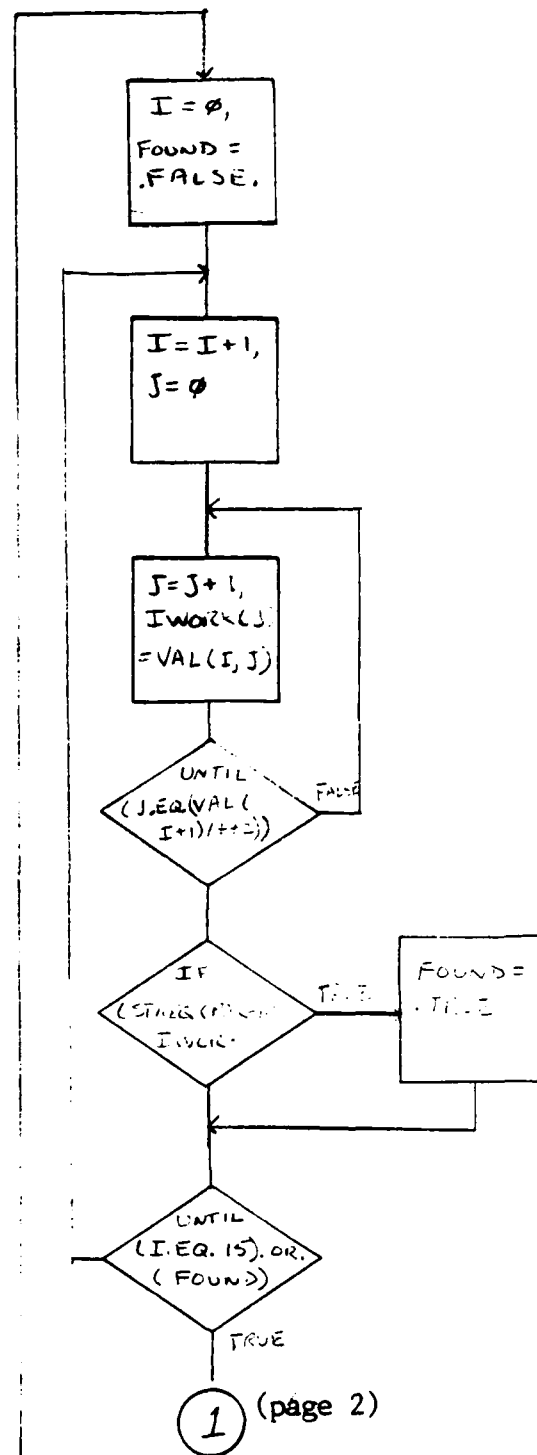
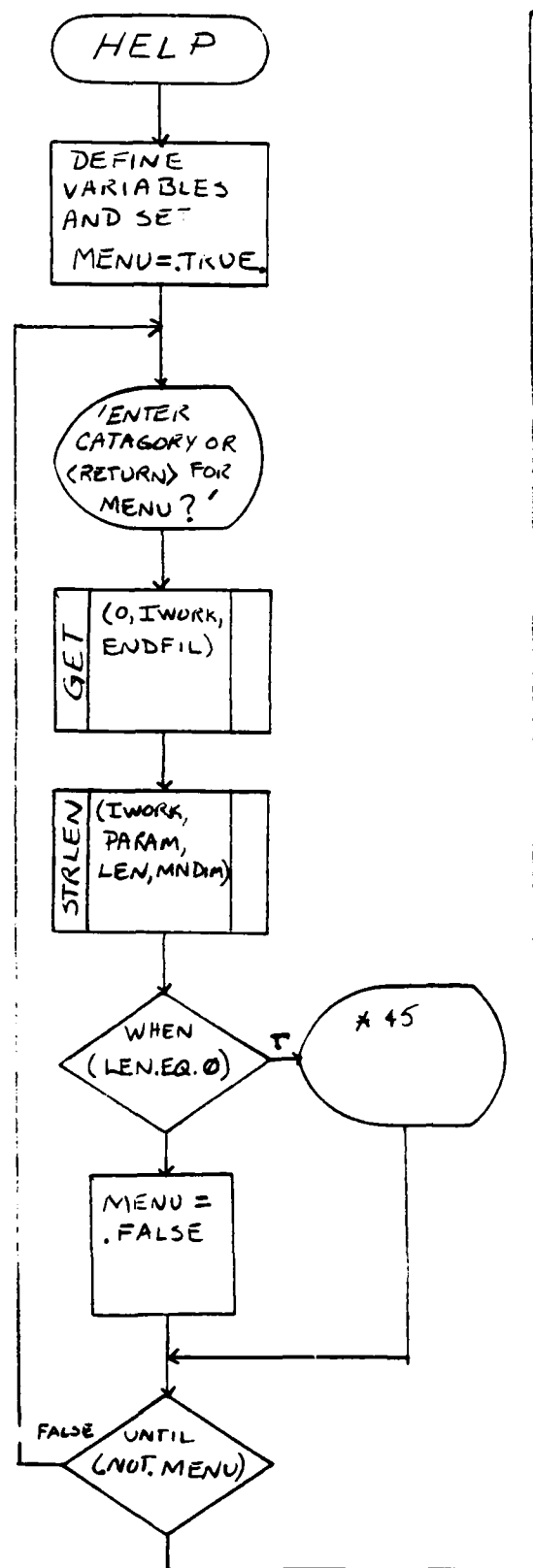
SUBROUTINE: HELP.FLC

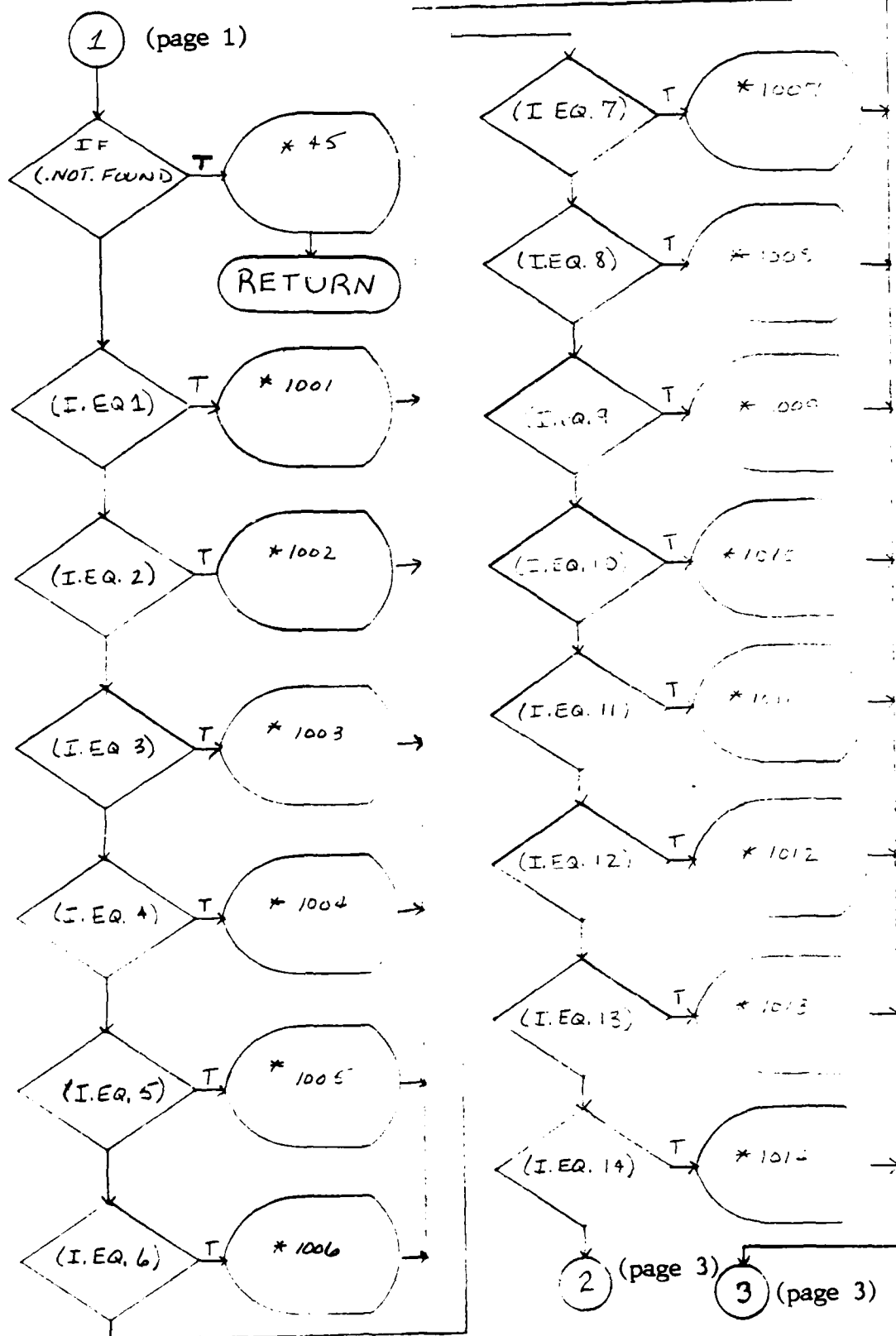
SYNTAX:

CALL HELP

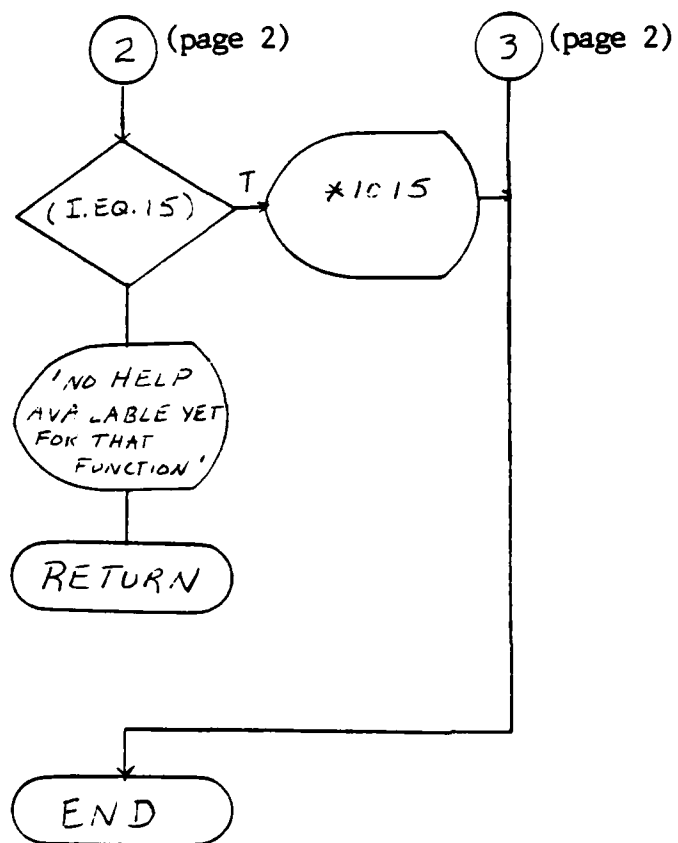
THIS IS A SUBROUTINE ADDED TO FEASIL 77 TO AID THE USER IN MANIPULATION OF DATA RELATIONS UNDER FEASIL 77. THE VALID PARAMETERS (PASSED IN THE INTEGER (HOLLERITH) ARRAY "PARAM") ARE:

CREATE	- EXPLAINS THE RELATION CREATION UTILITY.
EDIT	- EXPLAINS EDIT COMMAND STRUCTURES.
DELETE	- EXPLAINS THE DELETE UTILITY.
MERGE	- EXPLAINS THE MERGE RELATIONS PROCEDURE.
MODIFY	- EXPLAINS COLUMN MODIFICATION PROCEDURE.
RETRIEVE	- EXPLAINS RETRIEVE AND MANIPULATION UTILITY.
REORGANIZE	- EXPLAINS NEED FOR REORGANIZATION.
STATUS	- EXPLAINS THE STATUS RELATION COMMAND.
BACKUP	- EXPLAINS THE BACKUP UTILITY.
COLUMN	- EXPLAINS RELATION COLUMN STRUCTURES.
RECORD	- EXPLAINS RELATION RECORD STRUCTURES.
VOLUME	- EXPLAINS THE SYSTEM VOLUME AND IT'S USE.
PROTECTION	- EXPLAINS USE AND NEED FOR PROTECTION KEYS.
DEVICE	- EXPLAINS SYSTEM DEVICE STRUCTURE.
STRATEGY	- EXPLAINS FOUR TYPES OF DATA STRATEGIES.









**\*45 THE VALID HELP PARAMETERS ARE:**

BACKUP	CREATE
DELETE	EDIT
MERGE	MODIFY
REORGANIZE	RETRIEVE
STATUS	COLUMN
RECORD	VOLUME
PROTECTION	DEVICE
STRATEGY	

- \*1001** "CREATE" IS USED TO CREATE THE FILES ASSOCIATED WITH EVERY FEASIL RELATION. TO CREATE A FILE, YOU MUST KNOW HOW MANY COLUMNS THE RELATION REQUIRES, THE DESIRED PROTECTION KEYS ASSOCIATED WITH THE RELATION (IF DESIRED), AND THE COLUMN STRATEGIES FOR EACH OF THE COLUMNS CREATED. THE USER MUST ALSO KNOW THE NAME OF THE VOLUME ON WHICH THE RELATION IS TO BE CREATED. SEE HELP ON: PROTECTION, STRATEGY, COLUMN, VOLUME.
- \*1002** "EDIT" IS USED TO ENTER DATA INITIALLY OR TO MODIFY ALREADY EXISTING DATA. TO USE THE "EDIT" FEATURE, THE USER MUST KNOW THE NAME OF THE DESIRED RELATION, THE PROTECTION KEYS ASSOCIATED WITH THE RELATION, AND THE NAME OF THE VOLUME ON WHICH THE RELATION EXISTS. SEE HELP ON: PROTECTION, VOLUME.
- \*1003** "DELETE" IS USED TO PERMANENTLY REMOVE A RELATION FROM THE SPECIFIED VOLUME. TO USE THE "DELETE" FUNCTION, THE USER MUST KNOW THE RELATION NAME, THE PROTECTION KEYS, AND THE VOLUME NAME ON WHICH THE RELATION EXISTS. SEE HELP ON: PROTECTION, VOLUME.
- \*1004** "MODIFY" IS USED TO PERMANENTLY CHANGE THE:  
A> COLUMN NAME, OR  
B> NUMBER OF COLUMNS (DELETE OR ADD ONE).  
THE USER MAY ALSO LIST THE COLUMN NAMES USING THIS FEATURE.  
SEE HELP ON : COLUMN.
- \*1005** "REORGANIZE" IS USED TO INCREASE THE EFFICIENCY OF A FEASIL RELATION. IT IS PRIMARILY USED AFTER COLUMN SPECIFICATIONS HAVE BEEN MODIFIED (i. e., USING MODIFY). SEE HELP ON: COLUMN, MODIFY.

- \*1006 "RETRIEVE AND MANIPULATE" IS USED TO OPERATE ON AN EXISTING RELATION BY THE RELATIONAL DATA BASE OPERATORS (i. e., "AND", OR "OR"). IN ADDITION, R & M IS USED TO SORT RELATIONAL DATA BY COLUMNS, PRINT RELATIONAL DATA TO A DEVICE OR FILE, REPRODUCE A RELATION UNDER A DIFFERENT NAME, OR TO MOVE A SUBSET OF THE PRIMARY RELATION TO A NEW RELATION.  
SEE HELP ON: COLUMN, DEVICE.
- \*1007 "BACKUP" IS USED TO SAVE A BACKUP (COPY) OF THE RELATION TO (FROM) MAGNETIC TAPE. IN ADDITION TO STORING THE RELATIONAL DATA TO THE TAPE, THE DATE OF BACKUP AS WELL AS AN OPTIONAL PROTECTION KEY SEQUENCE ARE ALSO STORED.
- \*1008 "STATUS" IS USED TO INQUIRE ABOUT THE SIZE OF A RELATION AS WELL AS TO INQUIRE ABOUT THE "DEAD" SPACE USED BY THE FILE. THE "DEAD" SPACE MAY BE MINIMIZED, IF DESIRED FOR EFFICIENCY USING THE REORGANIZE FEATURE.  
SEE HELP ON : REORGANIZE.
- \*1009 "MERGE" SERVES TWO PURPOSES. IT'S FIRST PURPOSE IS TO PRODUCE A THIRD RELATION FROM TWO INPUT RELATIONS BY PERFORMING A "MERGE". THE MERGE FUNCTION IS A "UNION" OPERATOR. THEREFORE, THE OUTPUT RELATION CONSISTS OF THE TOTAL NUMBER OF UNIQUE COLUMNS IN THE TWO INPUT RELATIONS. THE ONLY RECORDS WHICH ARE MAINTAINED THROUGH THE "MERGE" ARE THE ONES WHICH INTERSECT ONE-TO ONE WITHIN THOSE COLUMNS WHOSE NAMES INTERSECT ONE-TO ONE.  
THE SECOND PURPOSE OF "MERGE" IS TO PRODUCE A THIRD RELATION FROM TWO INPUT RELATIONS WHICH HAVE THE IDENTICAL COLUMN NAMES AND STRATEGIES. THE RESULTING RELATION HAS THE SAME COLUMN NAMES AS THE TWO INPUTS AND THE RECORD ARE A "UNION" OF THE TWO INPUT RELATIONS.  
SEE HELP ON: COLUMN, RECORD, STRATEGY.
- \*1010 "COLUMN" IS THE NAME GIVEN TO EACH CATEGORY OF RELATION. EACH COLUMN HAS A UNIQUE COLUMN NAME AND MAY BE REFERENCED ALONE OR WITH OTHER COLUMNS IN THE SAME RELATION (IF THERE ARE ANY). AN EXAMPLE OF COLUMNS IN A RELATION CALLED "MYCLASS" MAY BE:
- 1 - NAME
  - 2 - AGE
  - 3 - SEX
  - 4 - SCORE

EACH RECORD OF THE RELATION HAS AN ENTRY FOR EACH COLUMN IN THE RELATION. THE RELATION, THUS, FORMS A DATABASE OF ENTRIES UNDER THE COLUMN HEADINGS.  
SEE HELP ON: RECORD.

\*1011 "RECORD" IS A COLLECTION OF COLUMNS WHICH, TOGETHER DESCRIBE THE DESIRED ASPECTS OF THE DATABASE. EACH RECORD IN A RELATION CONTAINS DATA FOR ALL OF THE COLUMNS OF THE RELATION. RECORDS MAY BE ACCESSED INDEPENDANTLY OR WITH OTHER RECORDS USING "RETRIEVE AND MANIPULATE".  
SEE HELP ON: COLUMN, RETRIEVE.

\*1012 "VOLUME" IS THE NAME GIVEN TO BULK MAGNETIC MATERIAL USED IN COMPUTER INFORMATION STORAGE. FOR EASY LOCATION OF DATA, THESE VOLUMES ARE UNIQUE NAMES. TYPICAL VOLUME NAMES ARE:

MT32  
MT6A  
PRO1  
TEMP

THE USER SHOULD INQUIRE ABOUT THE PROPER VOLUMN TO USE FOR EACH APPLICATION VIA THE SYSTEM OPERATOR.

\*1013 "PROTECTION". FEASIL -77 POSSESSES THE ABILITY TO OFFER A CERTAIN DEGREE OF PROTECTION AGAINST THE ACCIDENTAL DELETION OF RELATIONAL FILES. THOS SYSTEM OF PROTECTION INVOLVES THE USAGE OF PROTECTION "KEYS" ASSOCIATED WITH EACH RELATION. ON THE PERKIN-ELMER OS/32 SYSTEMS, THE READ/WRITE PROTECTION KEYS ARE CODED AS FOLLOWS:

CODE	DESCRIPTION
00	UNPROTECTED. ANY KEYS WILL WORK WHEN ACCESSING THIS RELATION.
10-254	CONDITIONALLY PROTECTED. THE USER MUST KNOW THE CORRECT CODES TO ACCESS THE RELATION.
255	UNCONDITIONALLY PROTECTED. NO KEYS WILL ALLOW ACCESS TO THE RELATION.

\*1014 "DEVICE" IS THE TERM USED TO IDENTIFY A NON-FILE PERIPHERAL ATTACHED TO THE COMPUTER TO/FROM WHICH DATA MAY BE SENT/RECEIVED. EXAMPLE DEVICES ARE:

PR: (SYSTEM PRINTER)

CON: (USER CONSOLE)

PLOT: (SYSTEM PLOTTER)

MAG0:(9 TRACK TAPE DRIVE)

CONTACT THE SYSTEM OPERATOR FOR MORE INFORMATION ABOUT THE SYSTEM'S PERIPHERAL DEVICES.

\*1015 "STRATEGY" IS THE TERM USED TO REFER TO THE TYPE OF DATA THAT IS CONTAINED IN A PARTICULAR COLUMN. THE VALID STRATEGIES ARE:

"1" - INTEGER NUMBER.

"2" - DECIMAL (REAL) NUMBER.

"3" - SINGLE CHARACTER, AND

"4" - CHARACTER STRING.

SEE HELP ON: COLUMN.

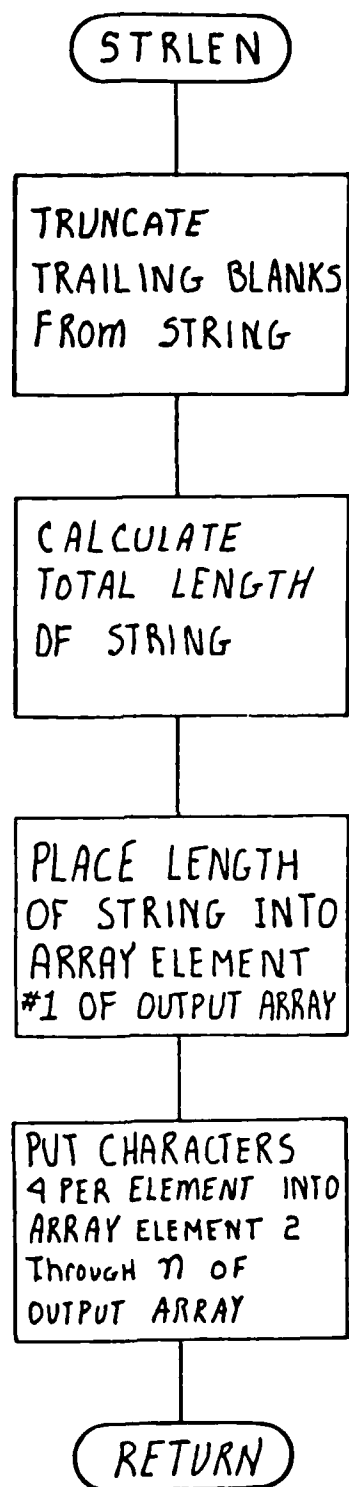
APPENDIX P

TPLIB1

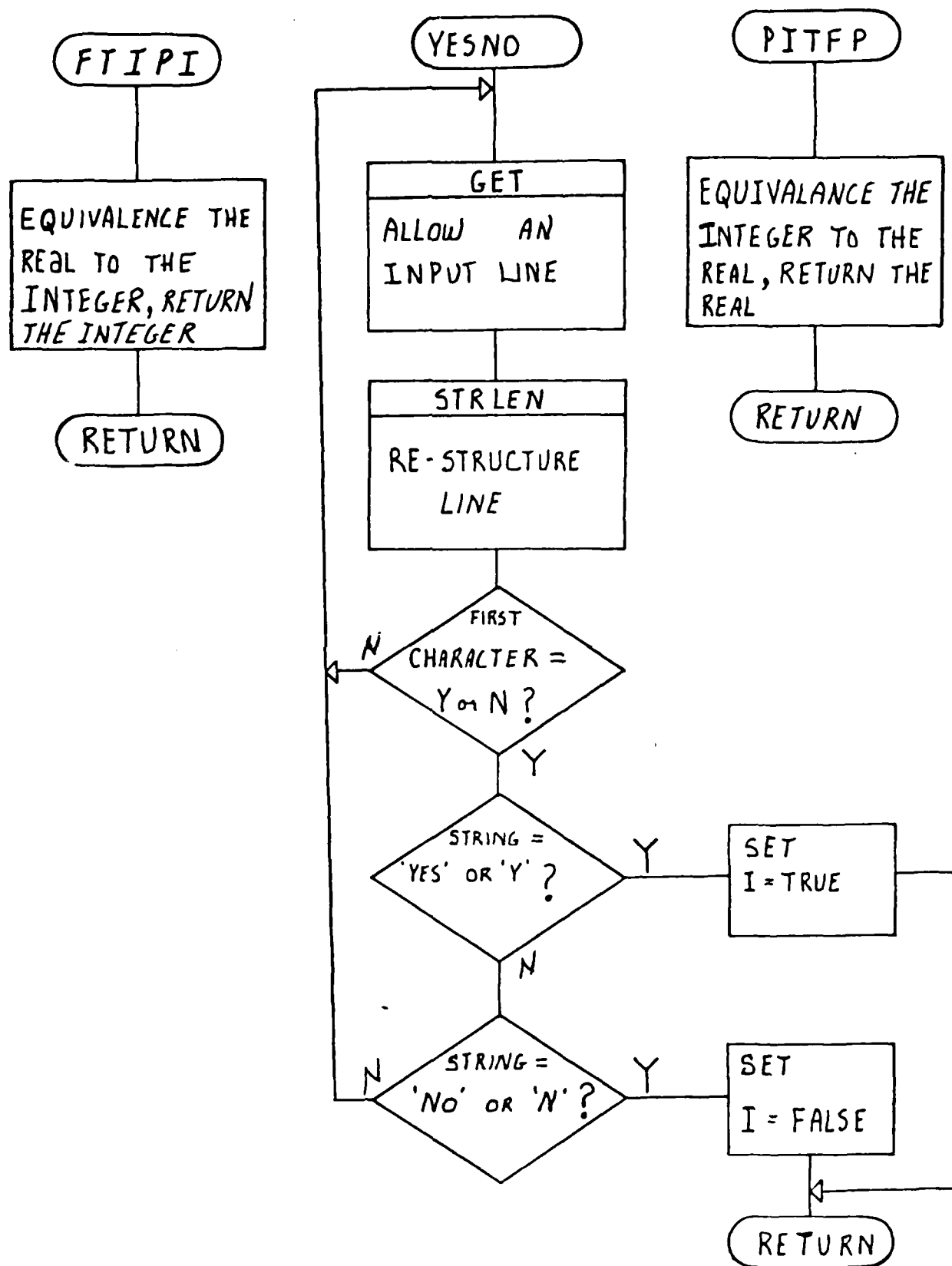
## Library: TPLIB1

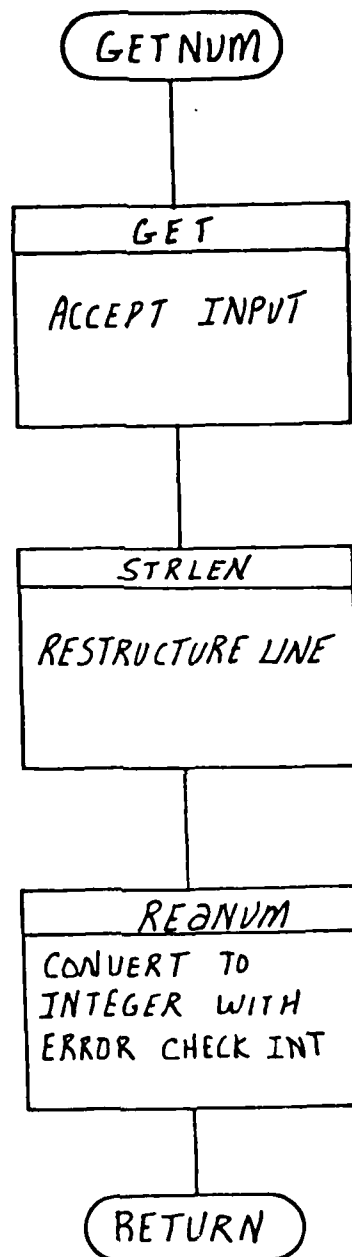
This library is a collection of subroutines used throughout FEASIL. A list of these subroutines and a brief description of their purpose is provided below.

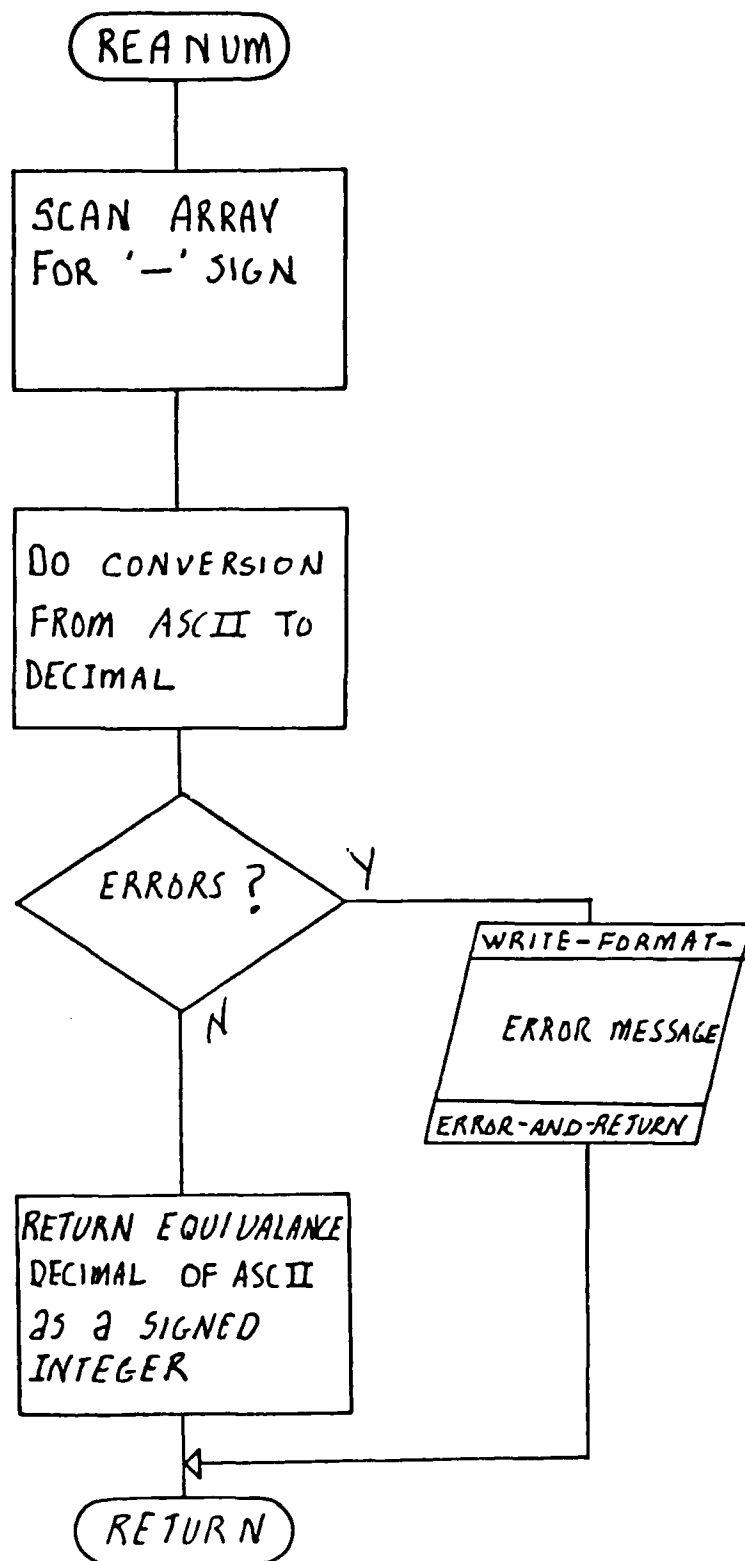
- STRLEN - Returns the length of a string.
- FTIPI - Converts floating point data to integer format.
- PITFP - Converts integer format data back to floating point.
- YESNO - Logical. Gets a "yes" or "no" answer from the console and returns true or false.
- GETNUM - Get integer data.
- REANUM - Converts a string into integer data.
- ELEB - Eliminates leading edge blanks.
- ENFPTA - Converts floating point data to ASCII format.
- ENINTA - Converts integer data into ASCII format.
- GETFP - Get floating point data.
- REAFP - Converts string data into floating point.

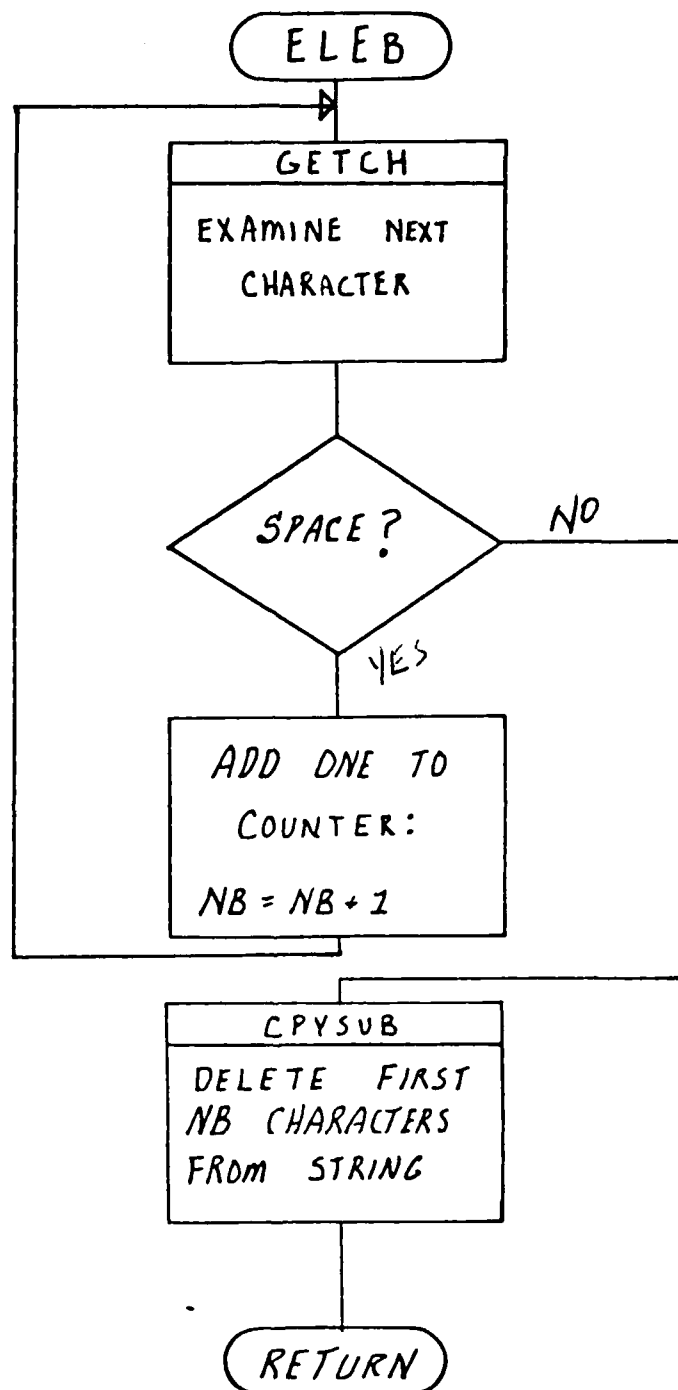


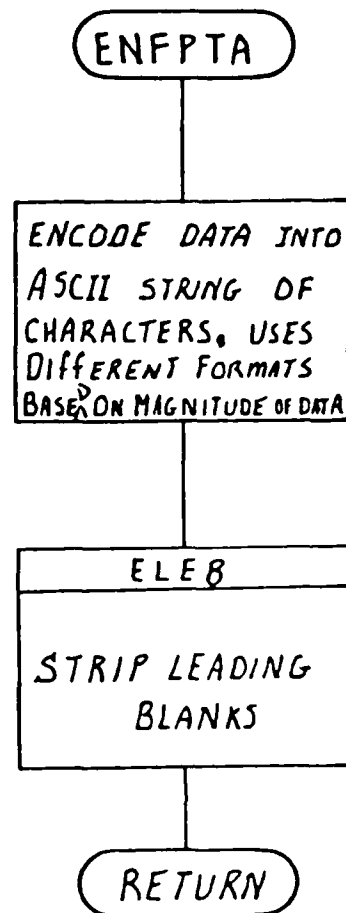
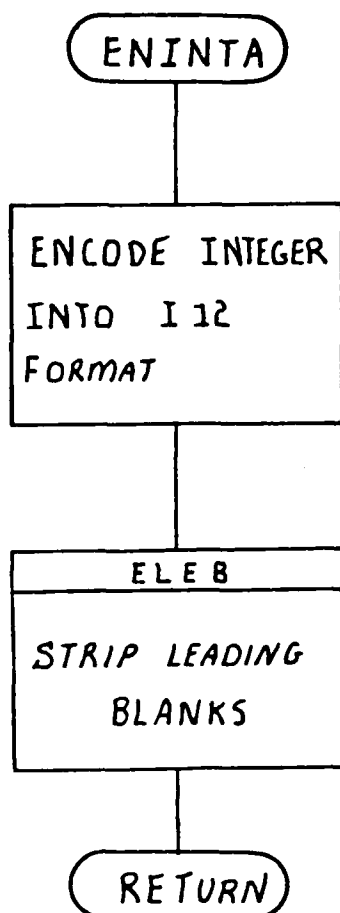


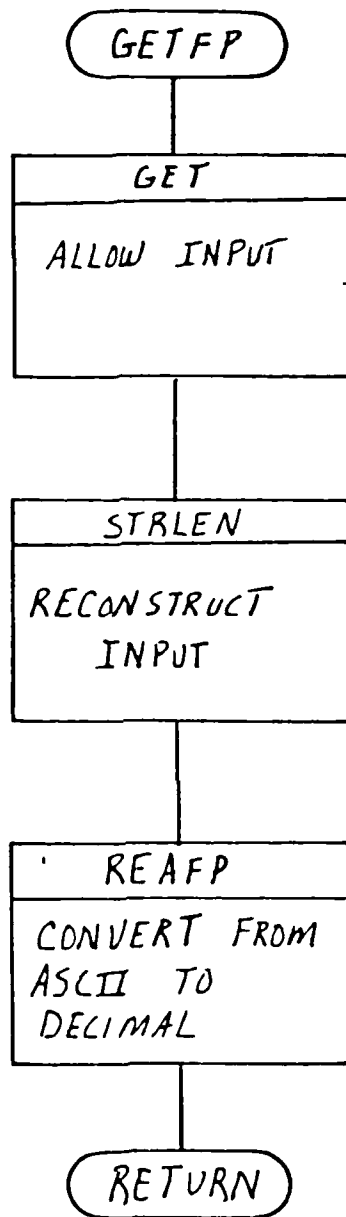


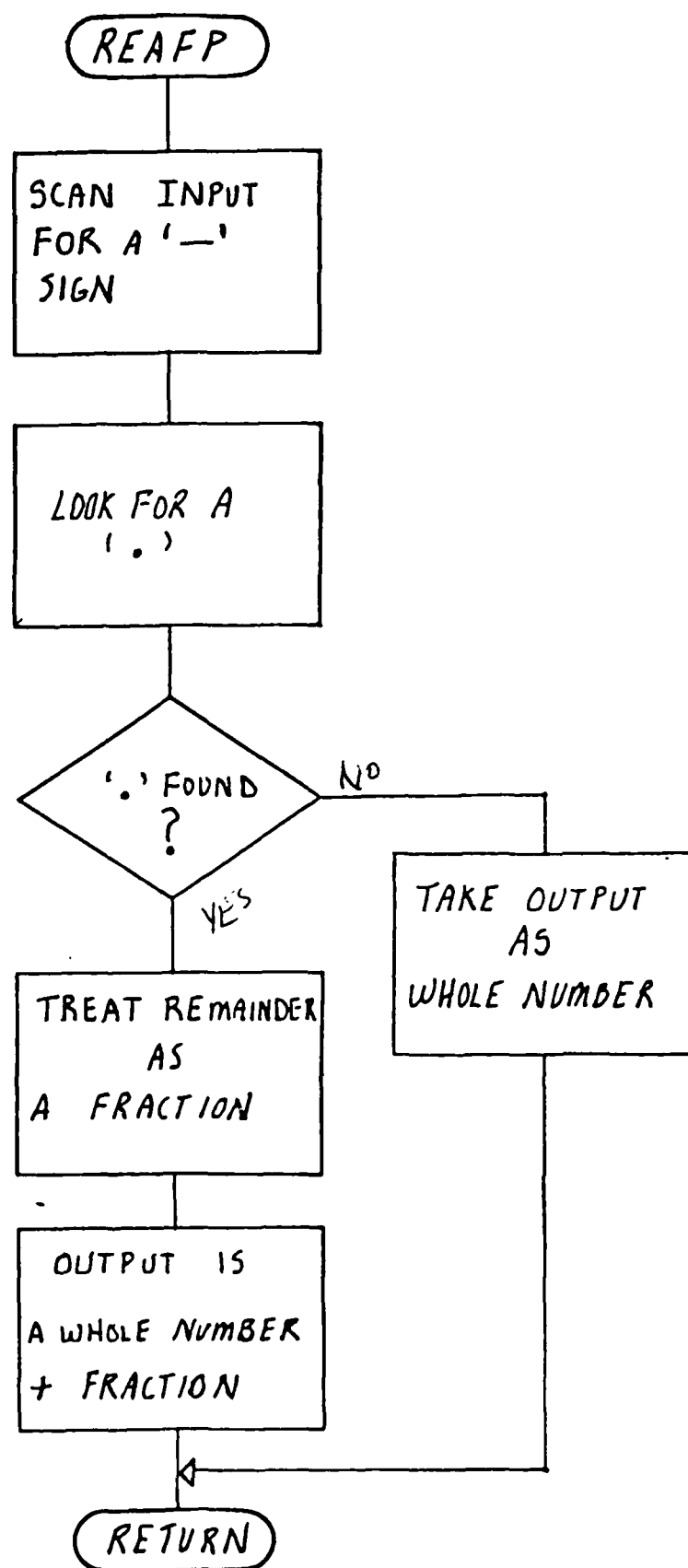












APPENDIX Q

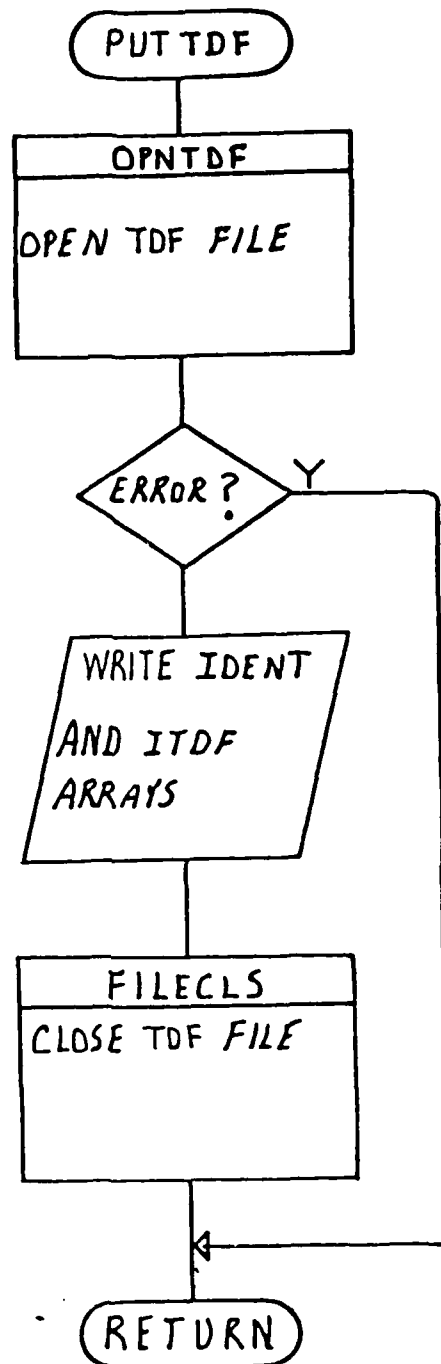
TPLIB2

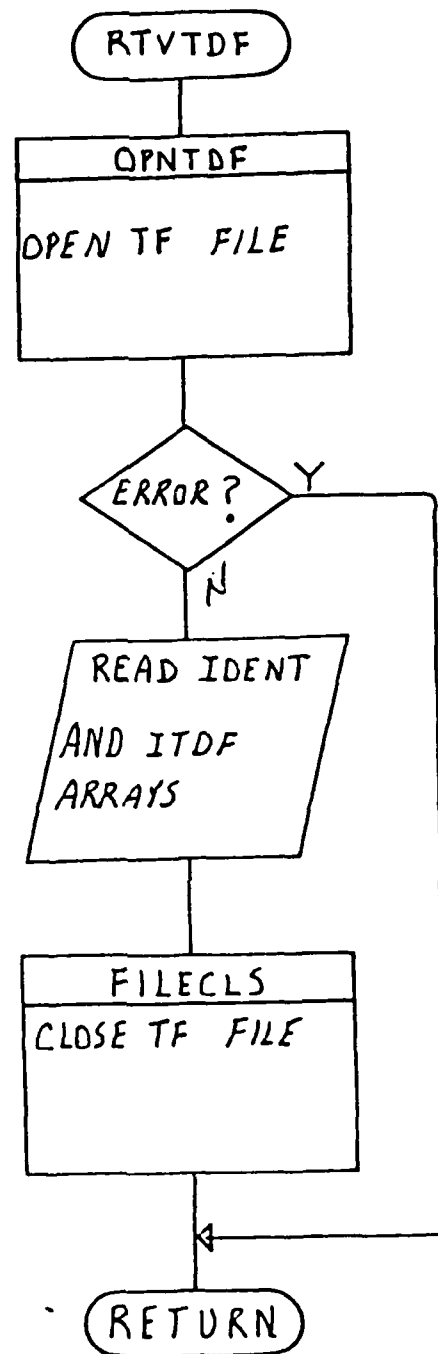


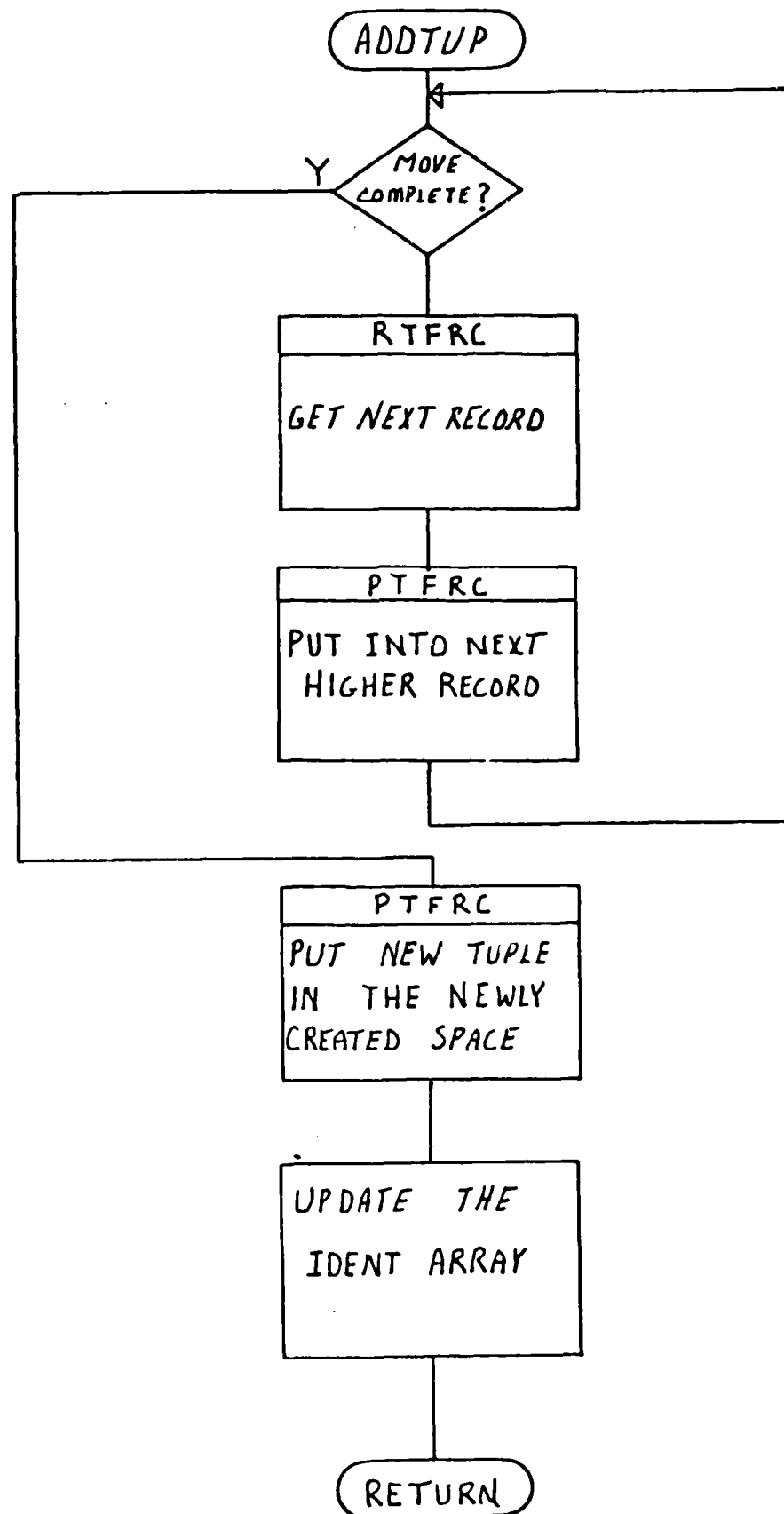
Library : TPLIB2

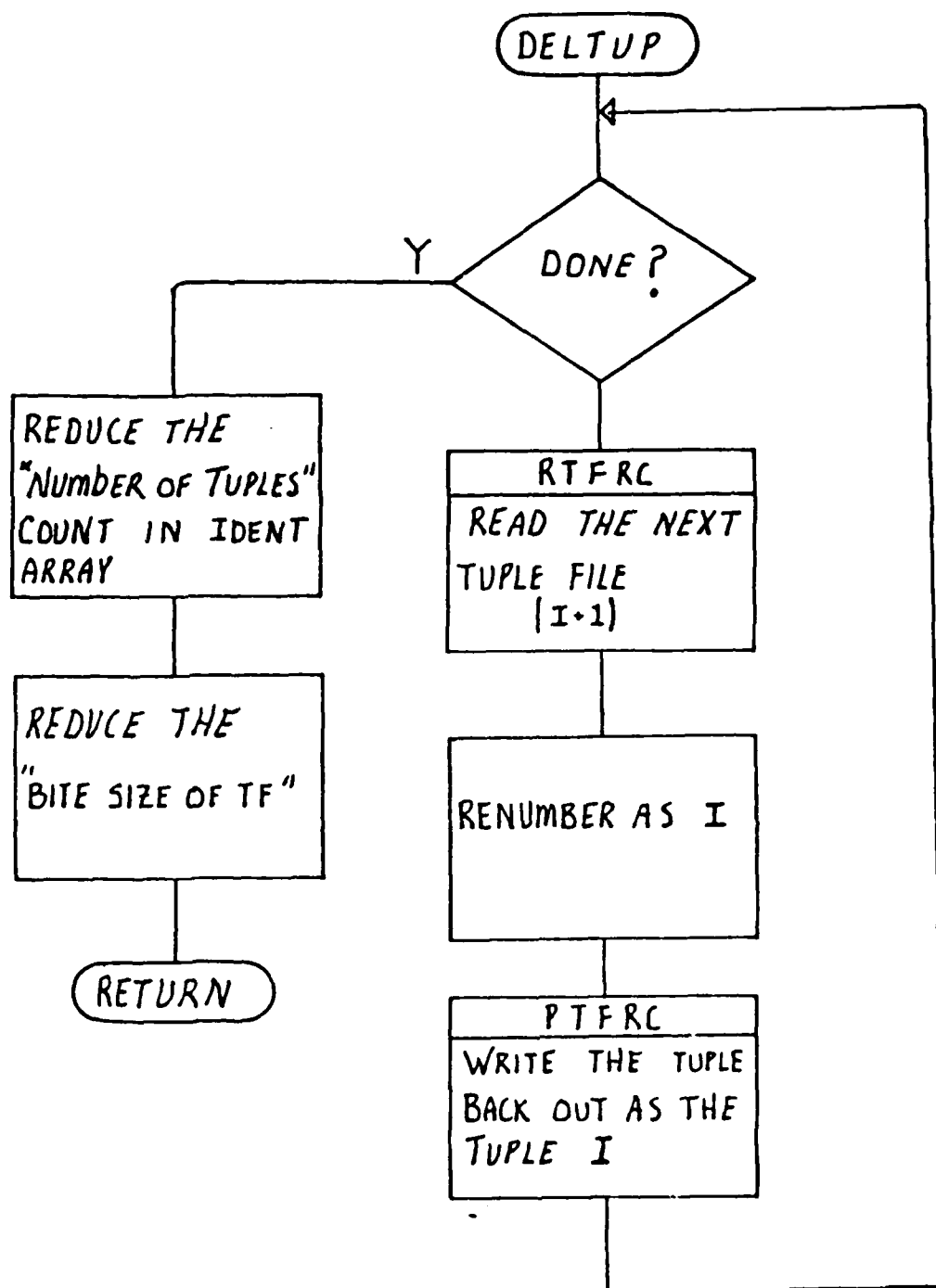
This library is a collection of subroutines used throughout FEASIL. A list of each of these subroutines and a brief description of their purposes is provided below:

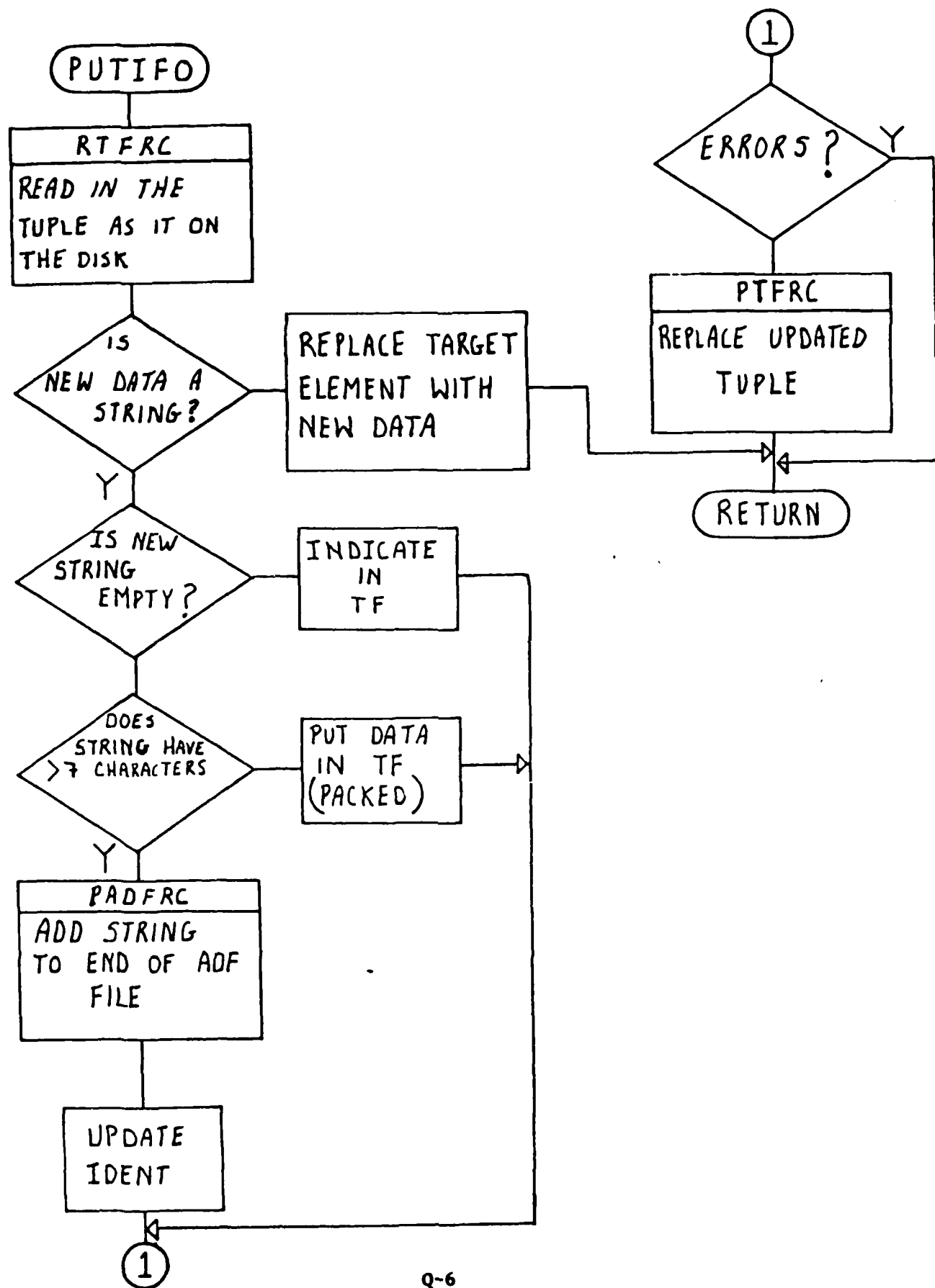
PUTTDF - Puts the IDENT and ITDF arrays on file.  
RTVTDF - Retrieves the IDENT and ITDF arrays from disk.  
ADDTUP - Adds tuple data to a relation.  
DELTUP - Deletes tuple data from a relation.  
PUTIFO - Put info into a relation.  
PTFRC - Puts data record into the tuple file.  
PADFRC - Puts data record into the Alpha Data File.  
RTFRC - Retrieves a record from the tuple file.  
RADFRC - Retrieves a record from the ADF.  
PUTIFO - Puts data into TF and ADF.  
COLNAM - Get the column names from the ADF  
HASHIT - Hashes a file name into 8 characters.  
UTFAR - Retrieves TF alpha record.

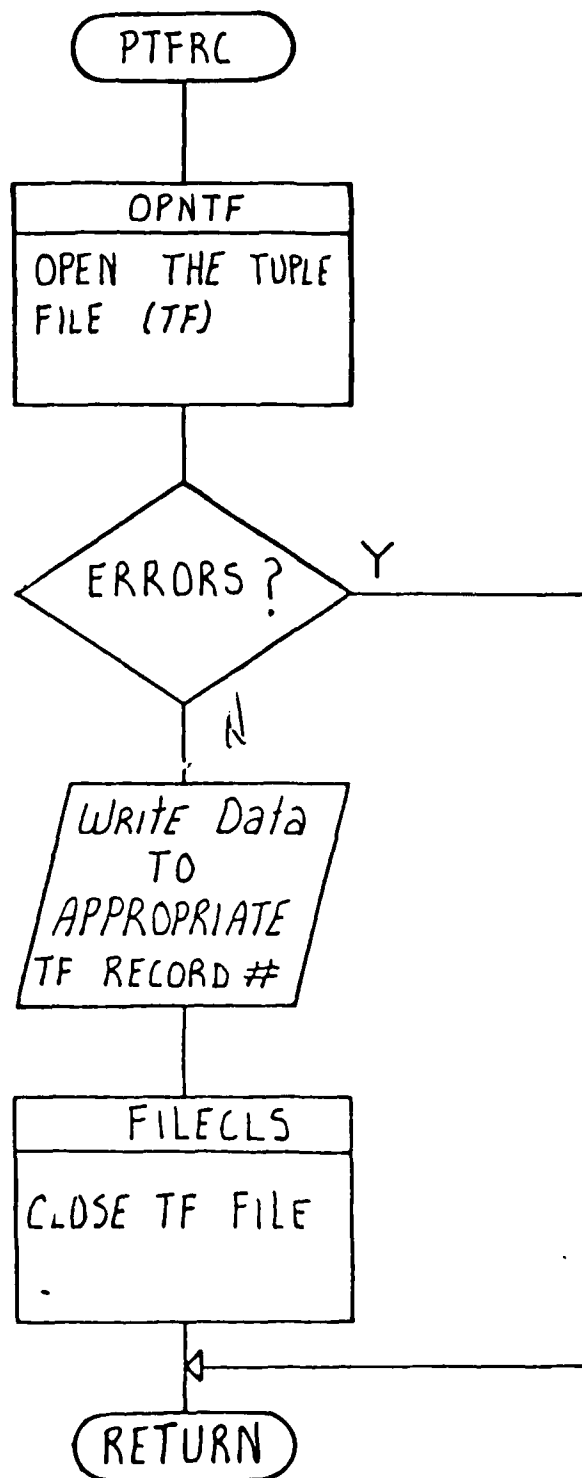


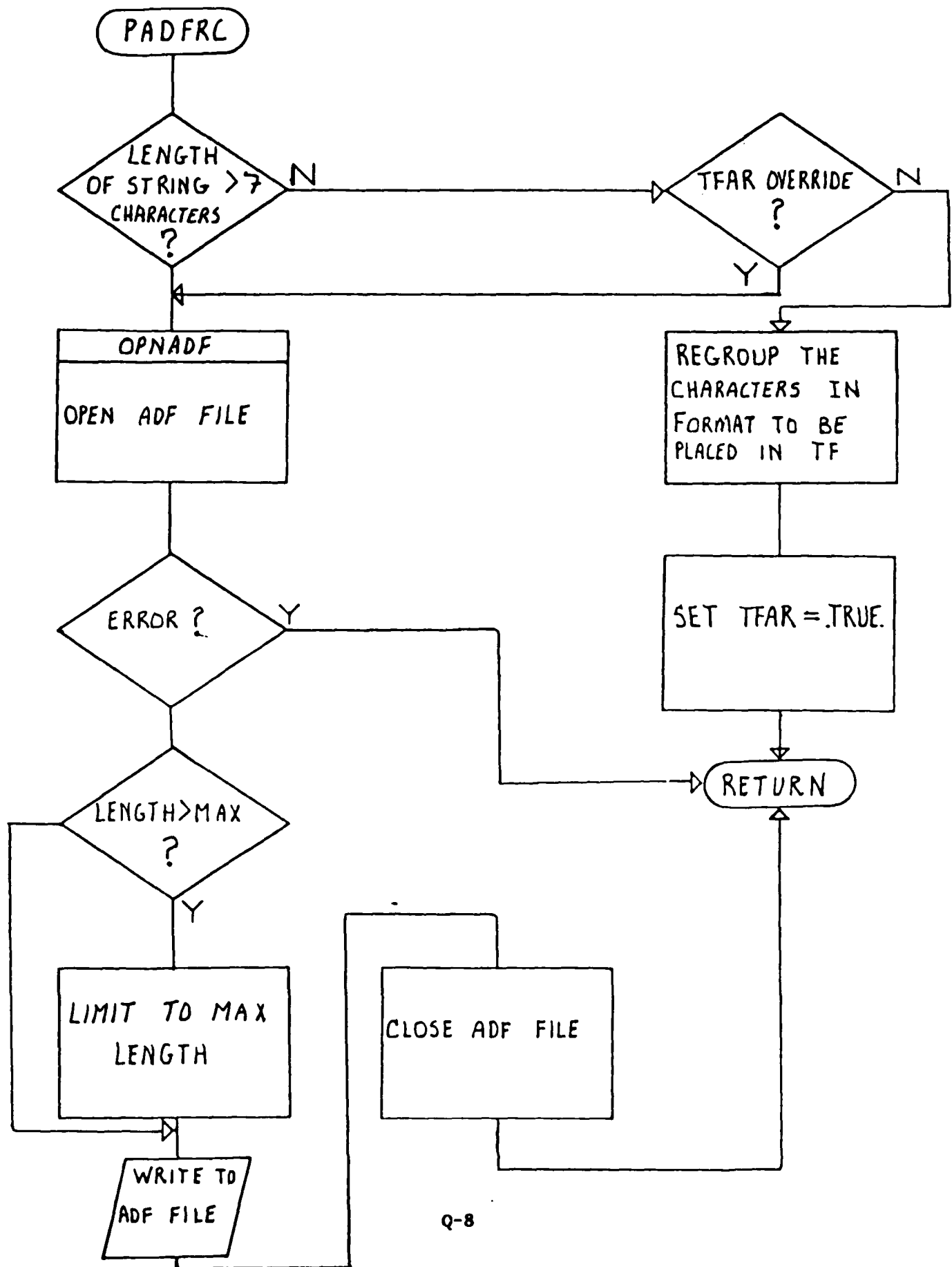




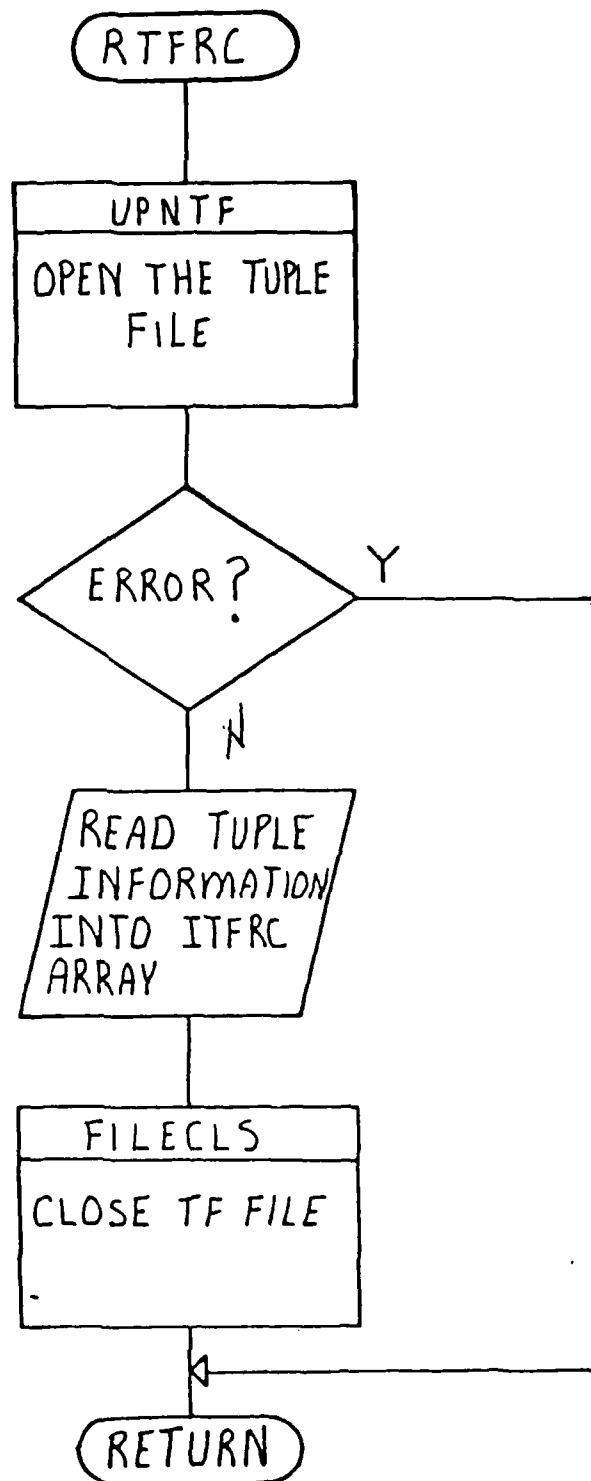


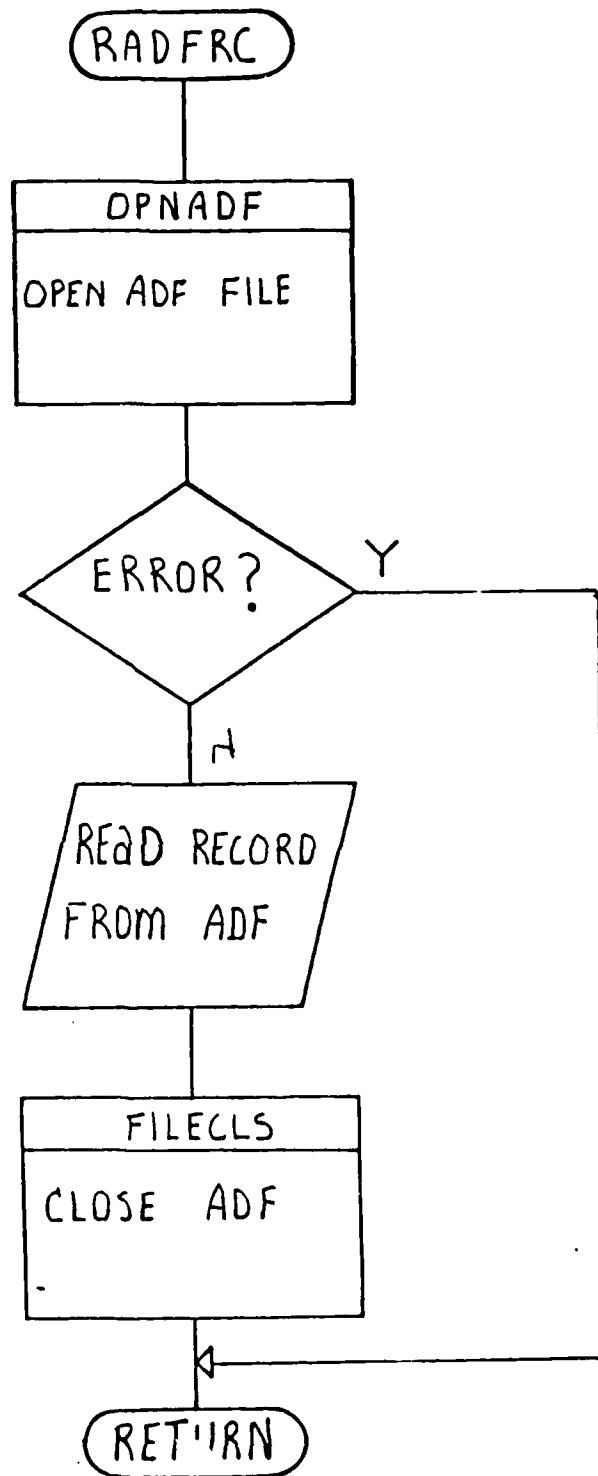


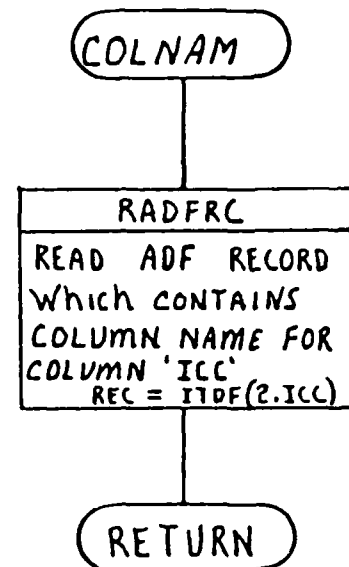
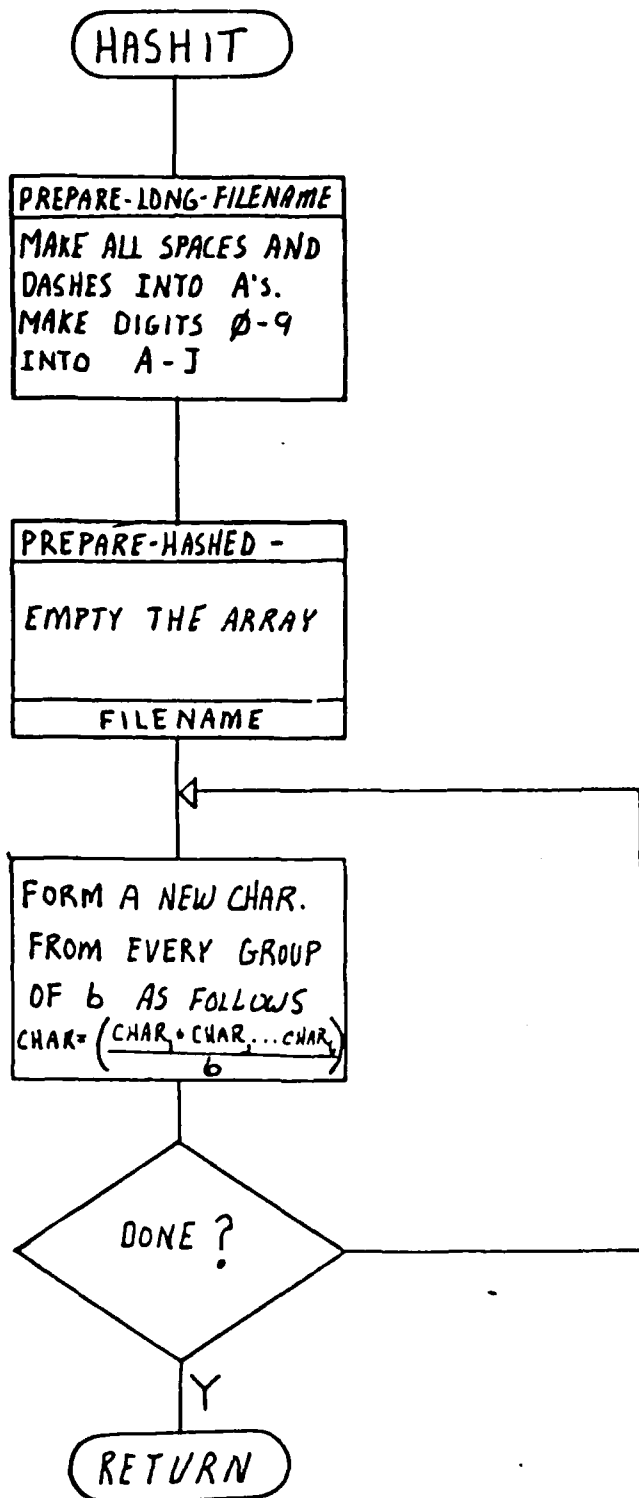


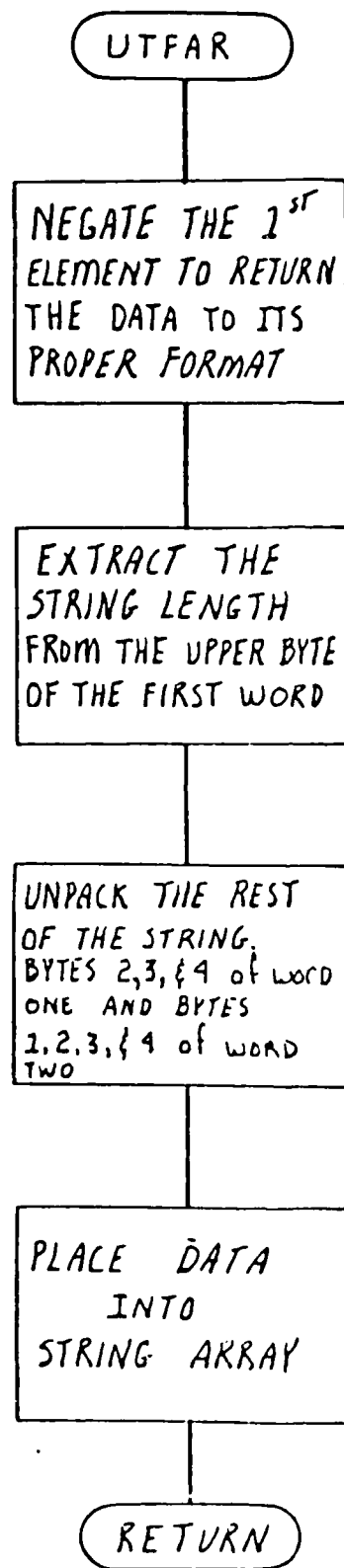












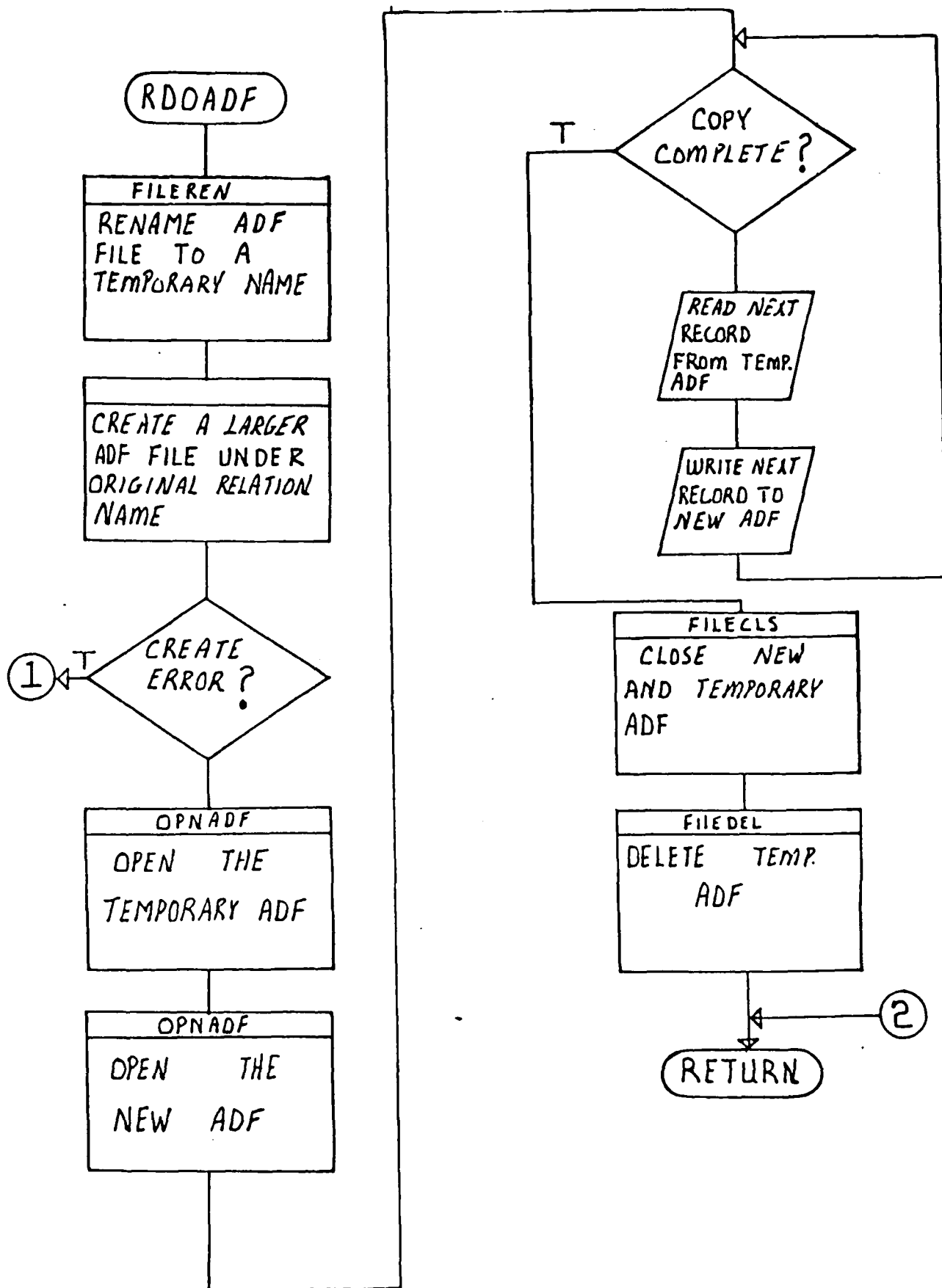
APPENDIX R

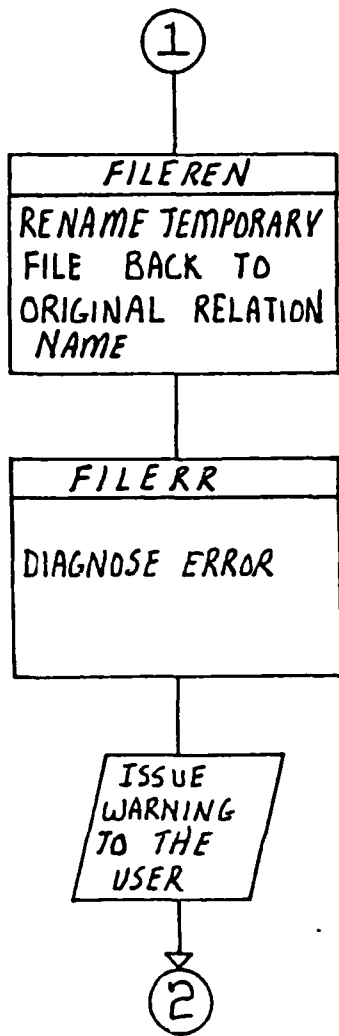
TPLIB3

Library: TPLIB3

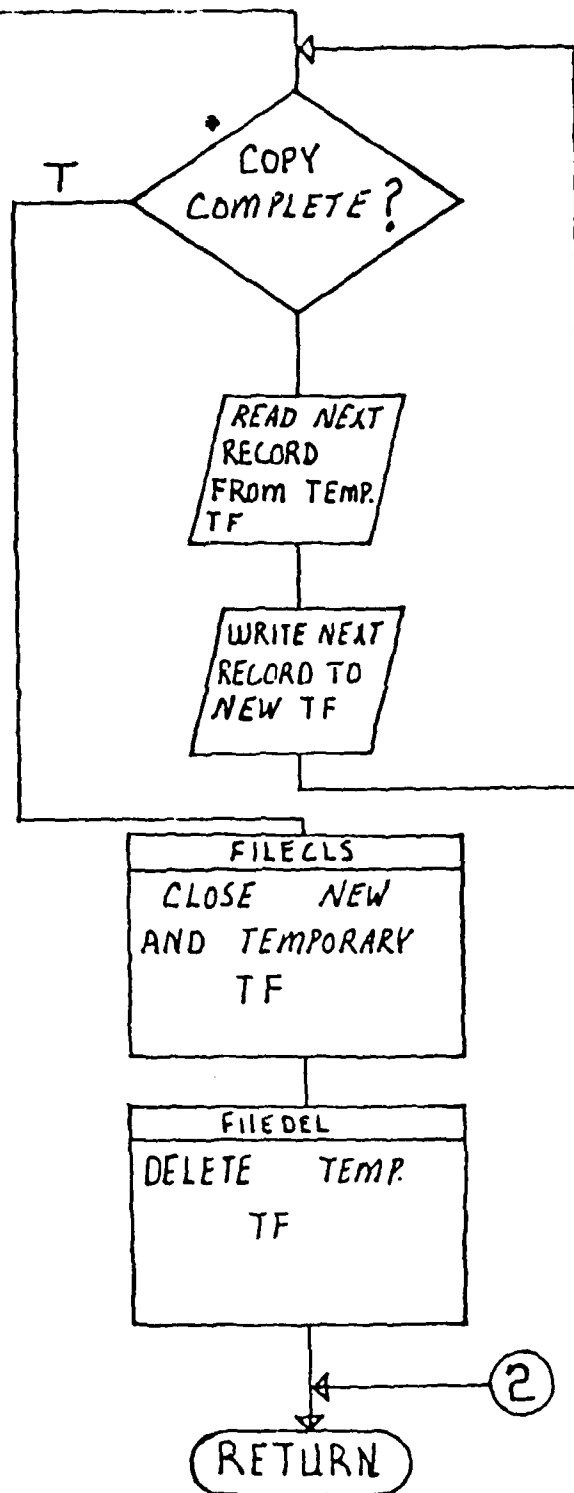
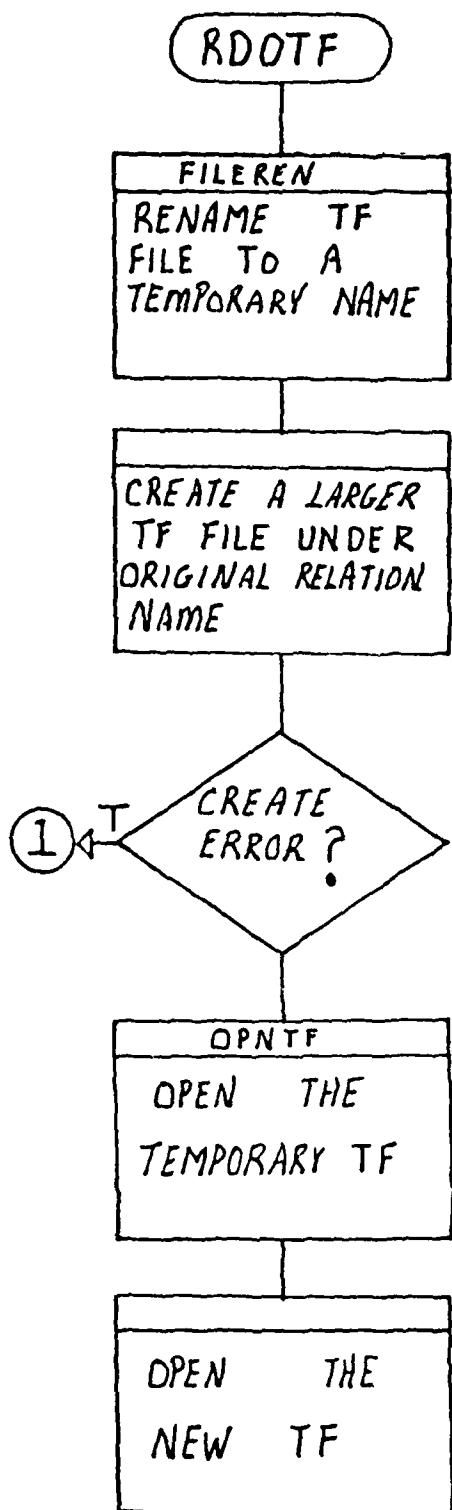
This is a collection of subroutines used for file reallocation and testing throughout FEASIL. A list of each subroutine and a brief description of their purpose is provided below.

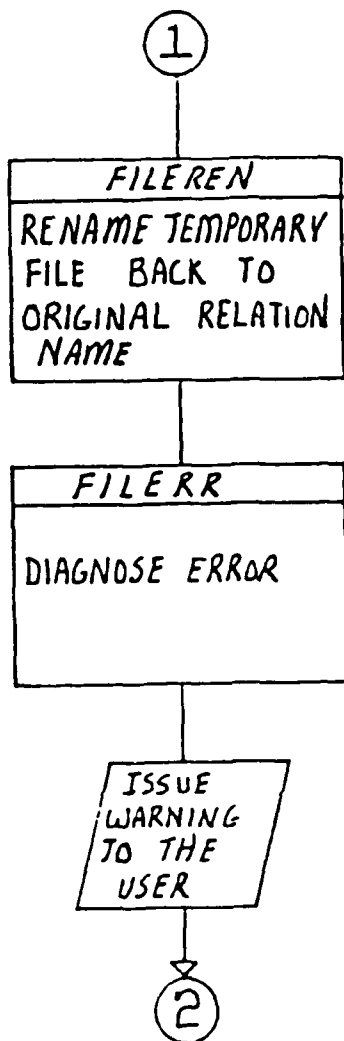
- RDOADF - Reallocates the size of the ADF.
- RDOTF - Reallocates the size of the TF.
- CPYREL - Copies the TF and ADF files.
- FSLIDE - Slides the first six elements of the argumnt one slot forward.

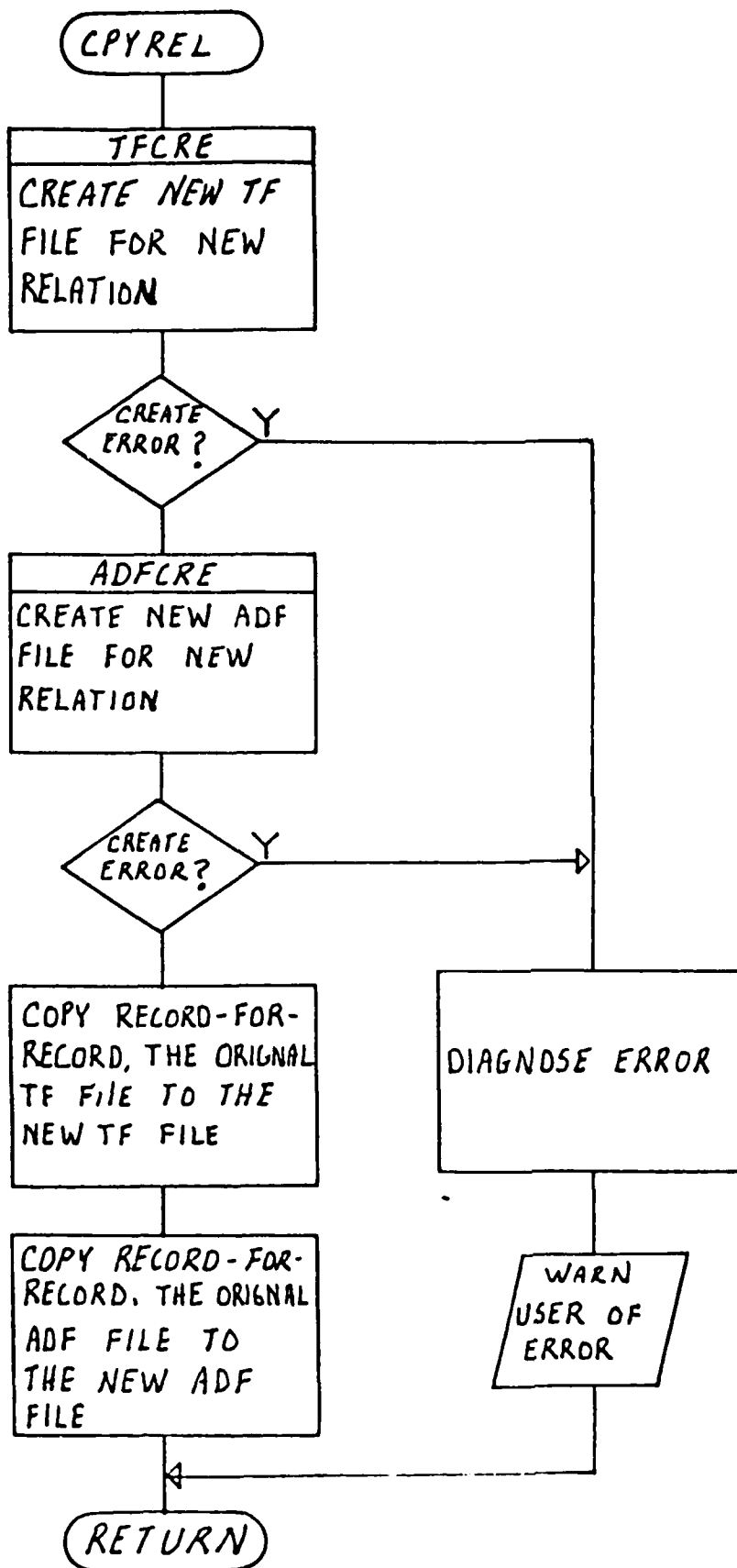


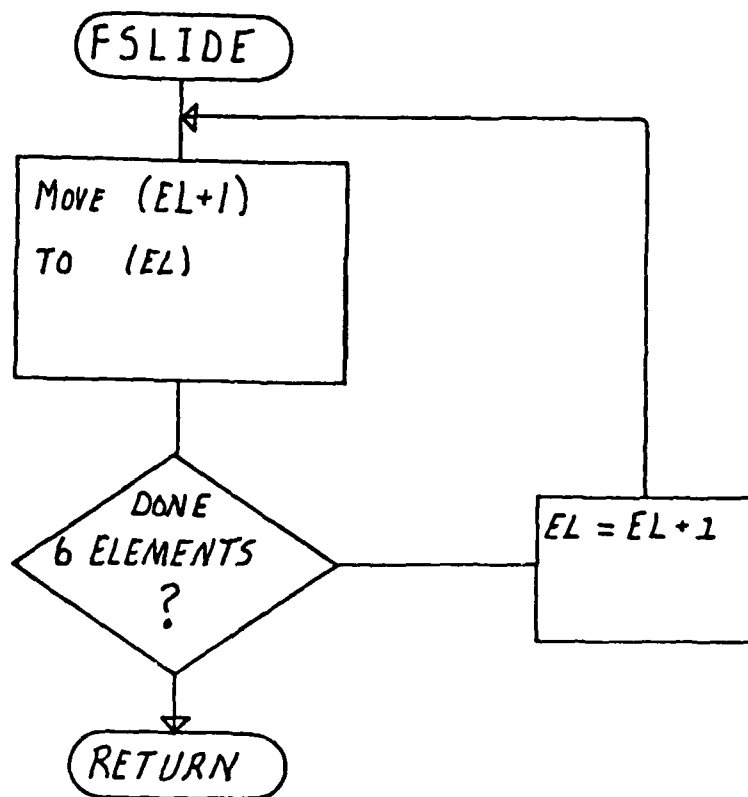












APPENDIX S

TPFILECR

OVERLAY: TPFILECR.FLC

The following is a complete list of subroutines included in the overlay TPFILECR.FLC with appropriate syntax for their use and a brief description of each routine. These routines will be described in detail on the following pages along with their flowcharts.

1. FILDES(TOTFD,NAMFIL)---creates file descriptor.
2. SECURE(IWKEY,IRKEY)---assigns security keys.
3. EMIARY(IARRAY,N)---empties the array "IARRAY" of size N.
4. TFCRE(TOTFD,IWKEY,IRKEY,NSZTR,ISTTF)---creates .TF file.
5. ADFCRE(TOTFD,IWKEY,IRKEY,NSZADF,ISTADF)---creates .ADF file.
6. TDFCRE(TOTFD,IWKEY,IRKEY,NSZTDF,ISTTDF)---creates .TDF file.
7. RELFC(TOTFD,NSZTF,NSZTDF,NSZADF,IFATAL)---creates relation.
8. TFDEL(TOTFD,IWKEY,IRKEY,ISTTF)---deletes .TF file.
9. TDFDEL(TOTFD,IWKEY,IRKEY,ISTTDF)---deletes .TDF file.
10. ADFDEL(TOTFD,IWKEY,IRKEY,ISTADF)---deletes .ADF file.
11. RELFD(IDELST,NAMFIL)---deletes relation
12. EQUIVI(ARRAY1,ARRAY2,N)---equates first N parts of 2 arrays.

FILES BY LOGICAL UNITS.

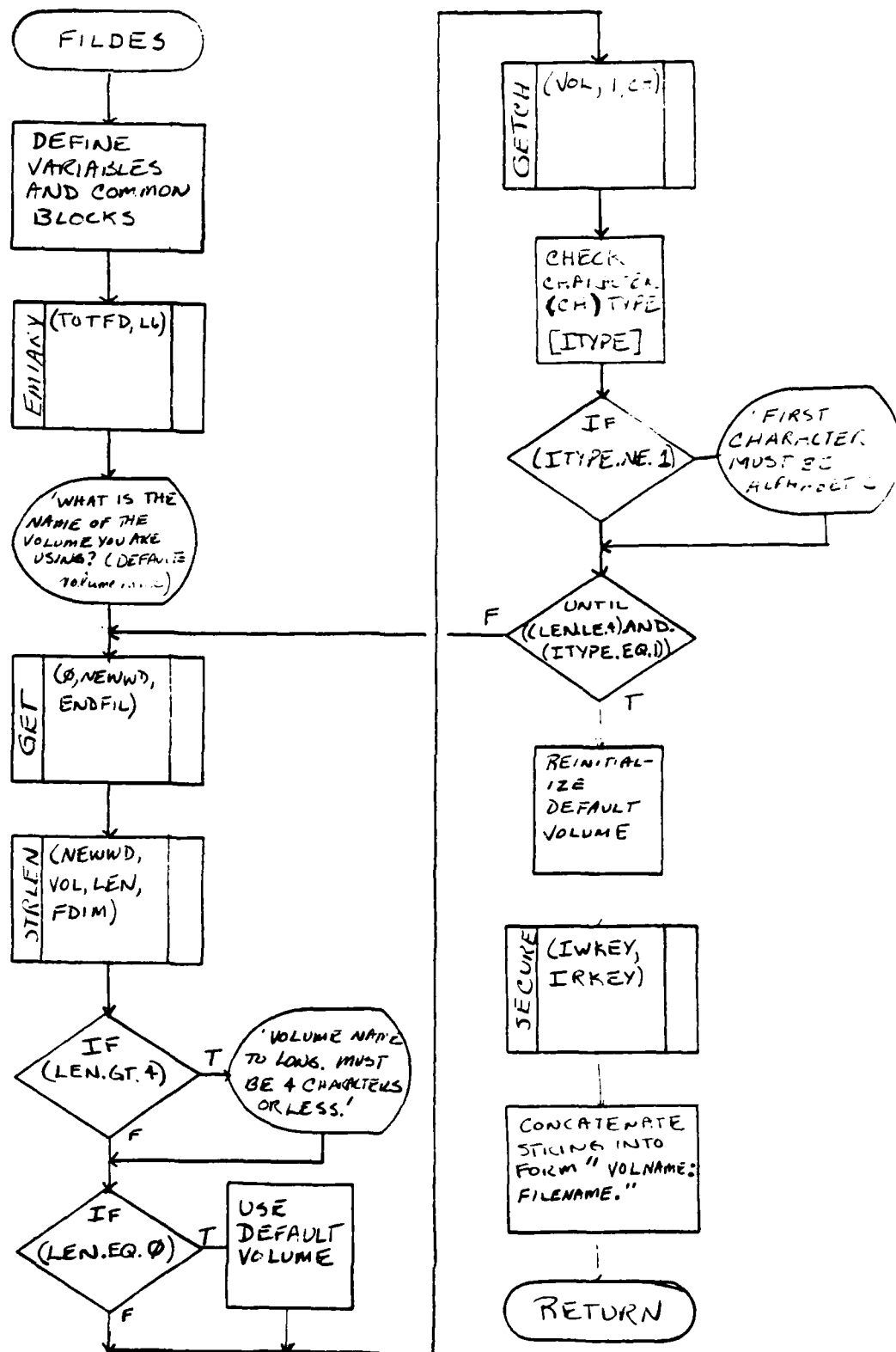
13. OPNTF(TOTFD,IWKEY,IRKEY,ISOTF,LUTF)---opens .TF file.
14. OPNADF(TOTFD,IWKEY,IRKEY,ISOADF,LUADF)---opens .ADF file.
15. OPNTDF(TOTFD,IWKEY,IRKEY,ISOTDF,LUTDF)---opens .TDF file.
16. FILERR(ISTATUS,NOMATCH)-- Diagnoses file I/O error codes.
17. NAME(I)---gets name "I".

SUBROUTINE: FILDES.FLC

SYNTAX: CALL FILDES(TOTFD,NAMFIL)

THIS ROUTINE INTERACTIVELY COMPLETES THE CONSTRUCTION OF THE FILE DESCRIPTOR, READ AND WRITE KEYS, AND STORAGE VOLUME.

NAMFIL IS AN INPUT NAME ARRAY DEFINED IN SUBROUTINE "NAME" AND TOTFD IS THE TOTAL FILE DESCRIPTOR FOR USE WITH STRING ROUTINES SUCH THAT: TOTFD - <VOLUME NAME>.<FILE NAME>.<EXT>.

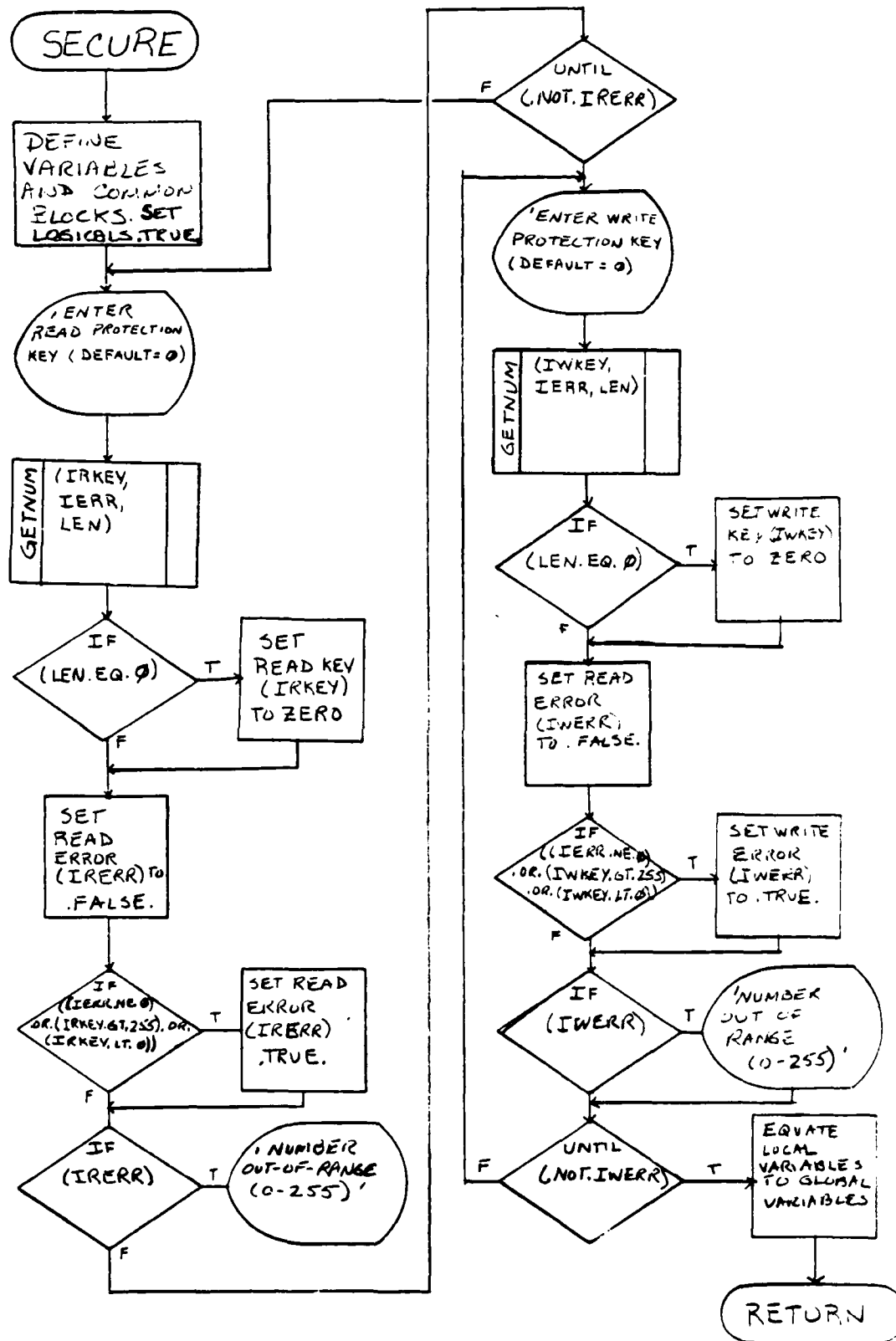




SUBROUTINE:SECURE.FLC

SYNTAX: CALL SECURE(IWKEY,IRKEY)

THIS ROUTINE OBTAINS THE SECURITY READ/WRITE KEYS FOR THE  
RELATION, IRKEY AND IWKEY, RESPECTFULLY.



SUBROUTINE: EMIARY.FLC

SYNTAX: CALL EMIARY(IARRAY,N)

THIS ROUTINE INITIALIZES AN INTEGER ARRAY NAMED "IARRAY" AND  
OF DIMENSION "N" TO EMPTY.

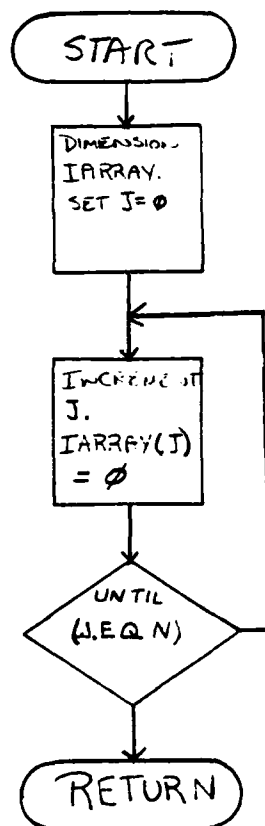
SUBROUTINES ADFCRE.FLC,TFCRE.FLC, AND TDFCRE.FLC

SYNTAX: CALL \*\*\*CRE(TOTFD,IWKEY,IRKEY,NSZ\*\*\*,IST\*\*\*)

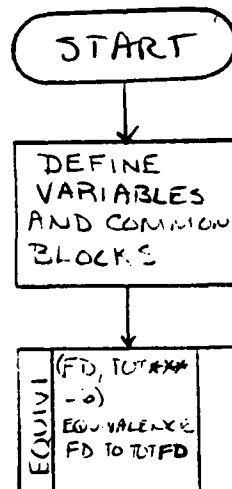
WHERE \*\*\* IS EITHER TF, TDF, OR ADF.

THIS ROUTINE ADDS THE EXTENSION \*\*\* TO THE TOTFD FILE TO  
CREATE THE TUPLE FILE (TF) DESCRIPTOR,THE ALPHA DATA FILE (ADF),  
OR THE TUPLE DESCRIPTOR FILE (TDF). THE APPROPRIATE FILE; EITHER  
TF, ADF, OR TDF,IS THEN CREATED AS A CONTIGUOUS FILE OF SIZE  
NSZ\*\*\* 256 BYTE SECTORS.

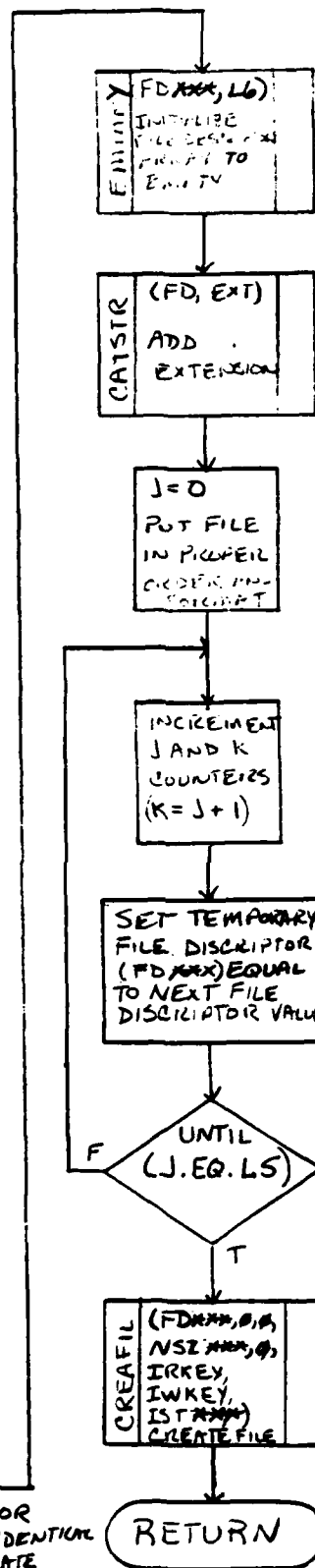
EMIARY (IARRAY, N)



TFCRE; ADFCRE; TDFCRE  
\*\*\*CRE (TDFD, IWKY, IRKY,  
NSZ\*\*\*, IST\*\*\*)



\*\*\* IS EITHER TF, ADF, OR  
TDF, THESE ARE THREE IDENTICAL  
SUBROUTINES USED TO CREATE  
THE APPROPRIATE FILES.

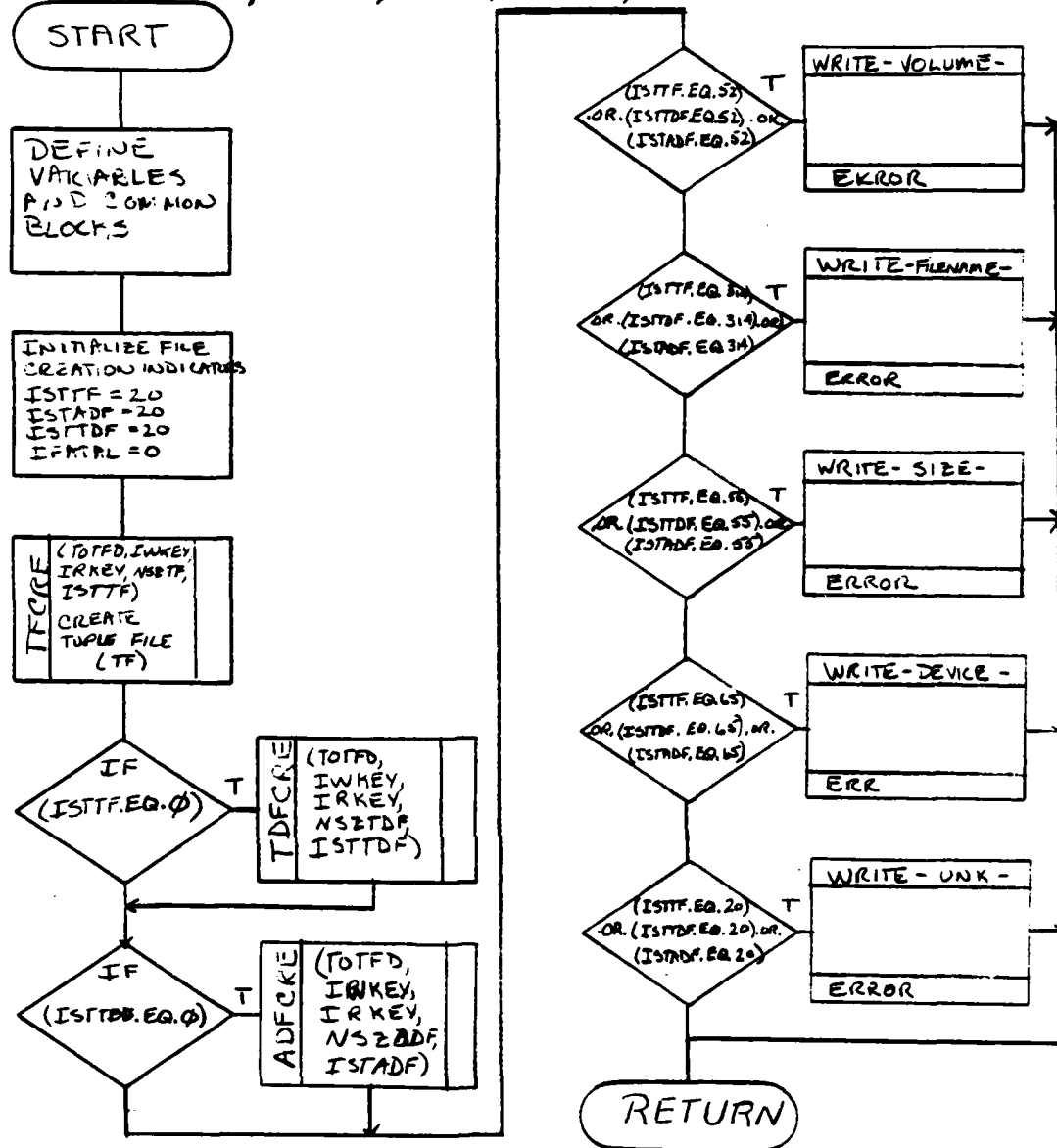


SUBROUTINE RELFC.FLC

SYNTAX: CALL RELFC(TOTFD,NSZTF,NSZTDF,NSZADF,IFATAL)

THIS ROUTINE IS TITLE THE "RELATIONAL FILE CREATOR" AND IT'S PURPOSE IS TO CREATE ALL DATABASE AND TEST MATRIX STORAGE FILES THE INITIAL TIME. THE VARIABLE "IFATAL" IS THE FILE CREATION RETURN STATUS INDICATOR. IF "IFATAL"-0 THEN THERE IS NO ERROR. IF IT EQUALS 1 THEN THERE IS NOT ENOUGH ROOM ON THE STORAGE VOLUME.

RELFC (TOTFD, NSZTF, NSZTDF, /NSZAE, IFATAL)



SUBROUTINES TFDEL.FLC, ADFDEL.FLC, AND TDFDEL.FLC

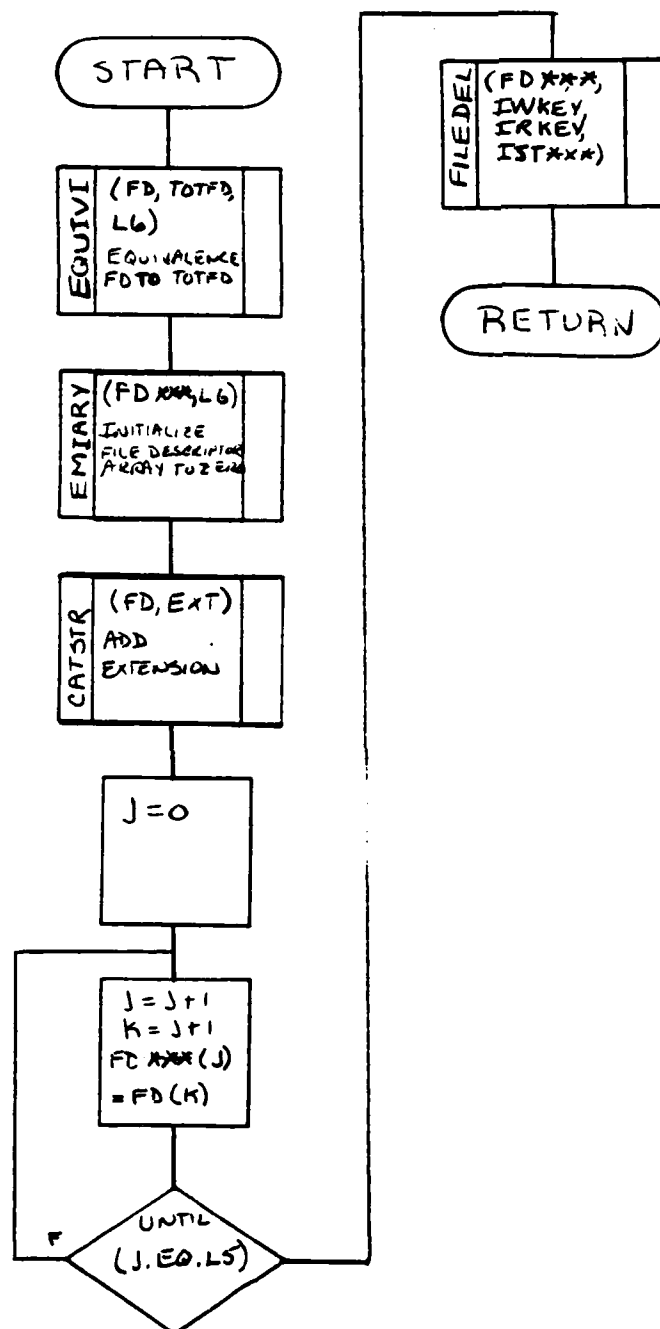
SYNTAX: CALL \*\*\*DEL(TOTFD,IWKEY,IRKEY,IST\*\*\*)

WHERE \*\*\* EQUALS EITHER TF, ADF, OR TDF.

THESE ROUTINES ADD THE EXTENSION \*\*\* TO THE TOTFD FILE TO  
CREATE THE TUPLE FILE (TF), THE ALPHA DATA FILE (ADF), OR THE  
TUPLE DESCRIPTOR FILE (TDF) DESCRIPTOR. THE APPROPRIATE FILE IS  
THEN DELETED.



TFDEL (TOTFD, IWKEY, IRKEY, ISTTF)  
 ADFDEL (TOTFD, IWKEY, IRKEY, ISTADF)  
 TDFDEL (TOTFD, IWKEY, IRKEY, ISTTDF)



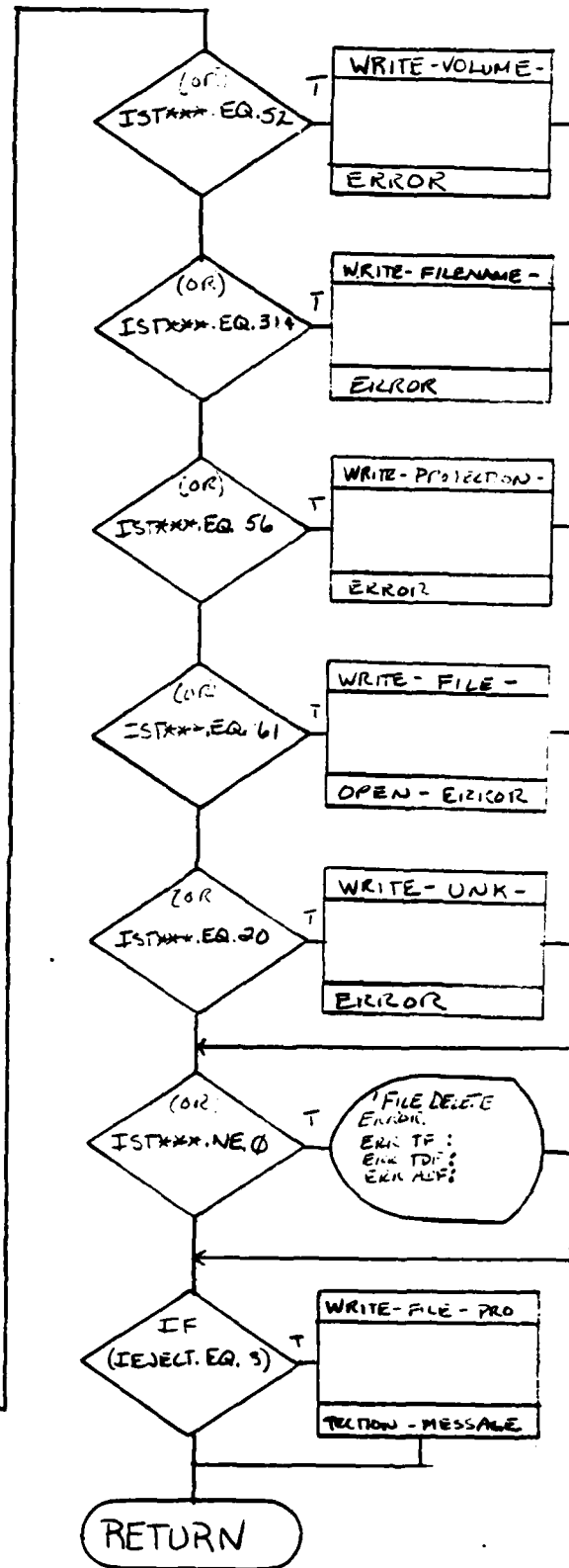
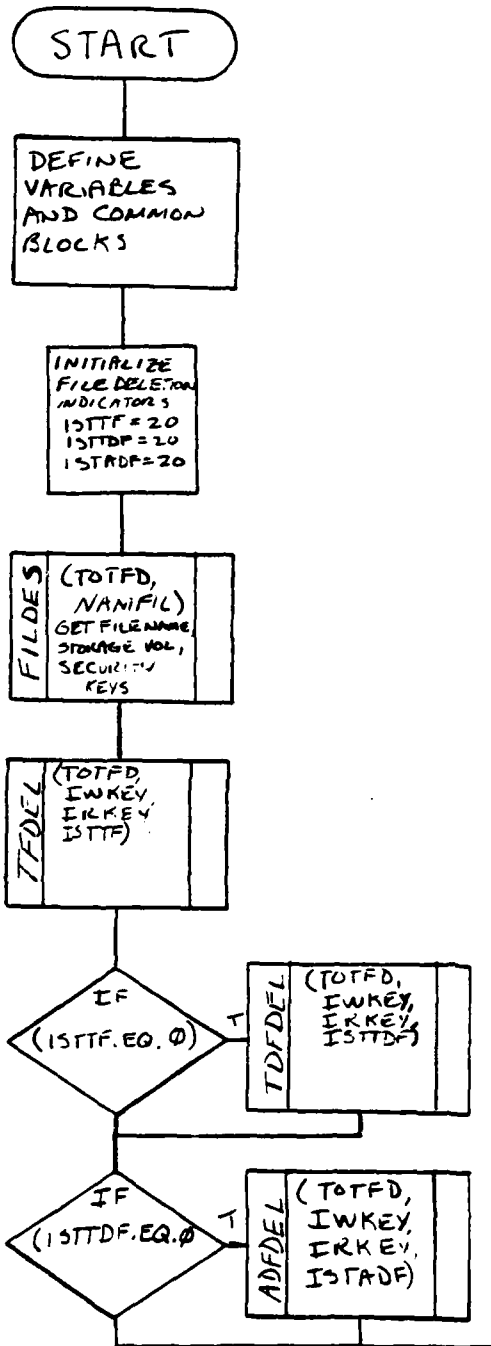
# SUBROUTINE RELFD.FLC

SYNTAX: CALL RELFD(IDELST,NAMFIL)

THIS ROUTINES TITLE IS "RELATIONAL FILES DELETER. IT DELETES ALL RELATIONAL DATABASE AND TEST MATRIX FILES ALL AT ONCE. "IDELST" IS A RETURN STATUS INDICATOR. IF IDELST EQUALS 0 THEN THERE IS NO ERROR. IF IDELST EQUALS 1 THEN THERE IS AN UNKNOWN ERROR. THE VARIABLE "NAMFIL" IS THE NAME OF THE RELATION TO BE DELETED. THE FORMAT FOR THIS VARIABLE IS DEFINED IN THE SUBROUTINE \* NAME \*.

THIS ROUTINE IS PRESET TO ALLOW THE USER THREE TRIES TO DELETE THE FILE. IF THE USER IS NOT SUCCESSFUL IT IS ASSUMED THAT HE EITHER DOES NOT KNOW WHAT HE IS DOING OR HE IS TRYING TO DELETE A FILE HE SHOULD NOT BE DELETING. AFTER THREE TRIES THE JOB IS TERMINATED.

RELFD (IDEALST 1-1-1-1)  
\*\*\* = .TF, .ADF, OR .TDF



SUBROUTINE EQUIVI.FLC

SYNTAX: CALL EQUIVI(ARRAY1,ARRAY2,N)

THIS ROUTINE EQUIVALENCES THE FIRST N ELEMENT OF INTEGER  
ARRAY1 TO INTEGER ARRAY2

SUBROUTINES OPNTF.FLC, OPNADF.FLC, AND OPNTDF.FLC.

SYNTAX: CALL OPN\*\*\*(TOTFD,IWKEY,IRKEY,ISO\*\*\*,LU\*\*\*)

WHERE \*\*\* EQUALS TF, ADF, OR TDF.

THESE ROUTINES OPEN THE TUPLE FILE (TF), THE ALPHA DATA FILE (ADF), OR THE TUPLE DESCRIPTOR FILE (TDF) FOR THE RELATION NAMED IN THE TOTFD.

\*\*\*\*\* WARNING \*\*\*\*\* WARNING \*\*\*\*\*

ALL FILES ARE OPENED AS SHAREABLE READ/WRITE  
(i.e., IAP-4)

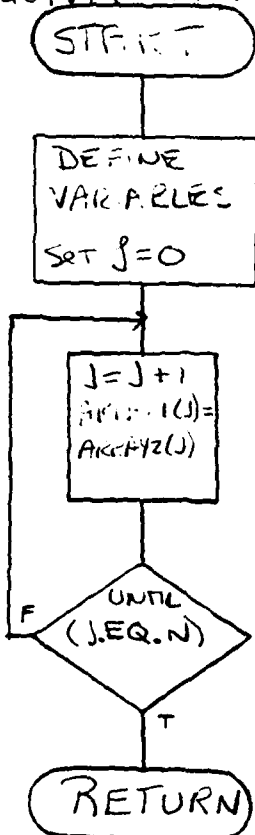
TF FILES ARE LOCATED ON LOGICAL UNIT LU-LUTF.

ADF FILES ARE LOCATED ON LOGICAL UNIT LU-LUADF.

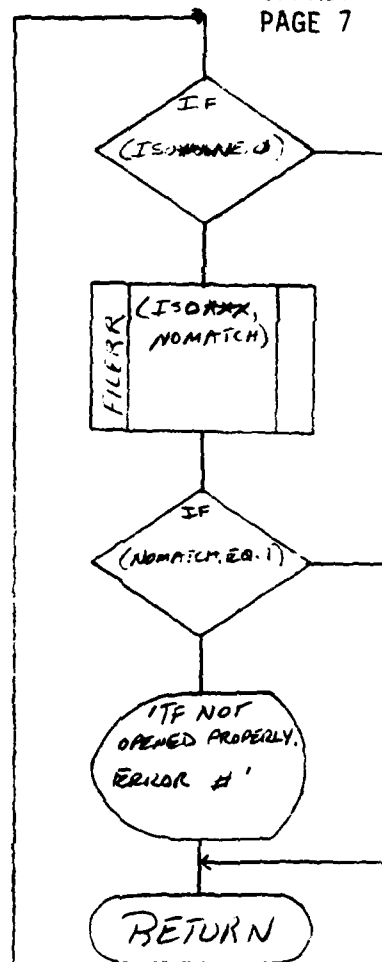
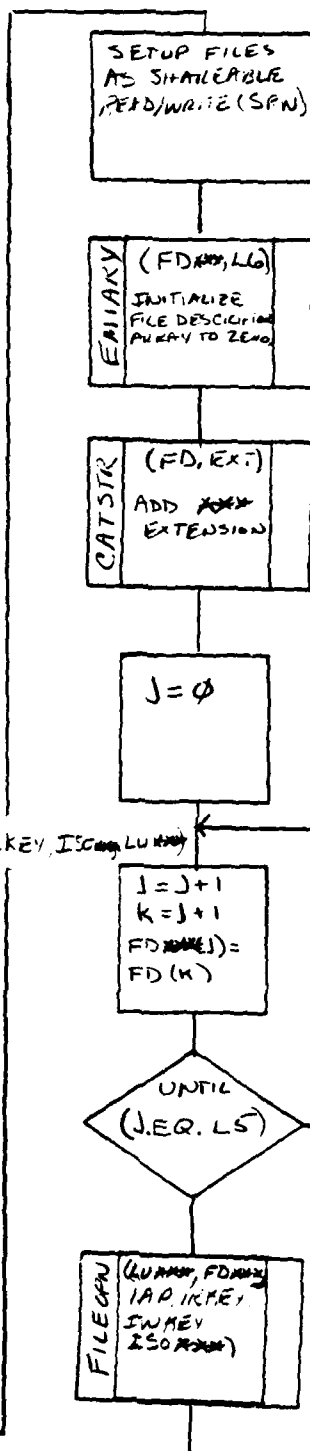
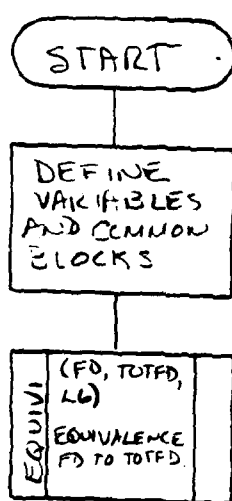
TDF FILES ARE LOCATED ON LOGICAL UNIT LU-LUTDF.

IST IS AN ERROR INDICATOR. FOR MORE INFORMATION REFER TO  
THE FTN LIB REAL TIME EXTENSION MANUAL.

EQUIVI (AFIAT, AFIAT, N)



OPN\*\*\* (TOTFD, IWKEY, ICKEY, ISO\*\*\*, LU\*\*\*)



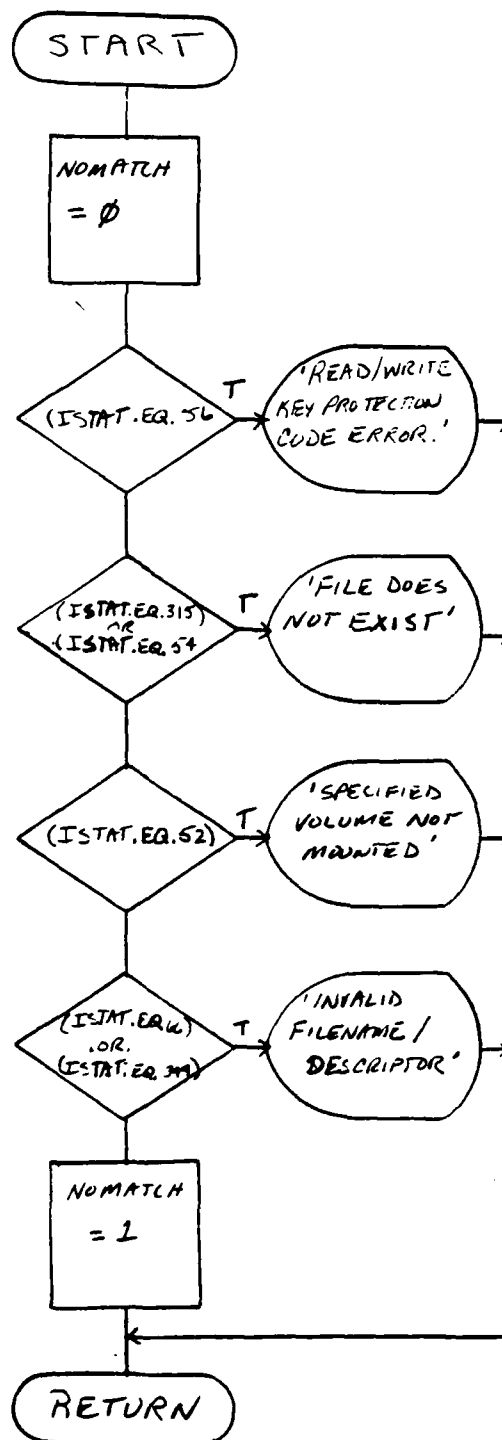
\*\*\* = TF, TDF, OR ADF.

SUBROUTINE FILERR.FLC

SYNTAX: CALL FILERR(ISTATUS,NOMATCH)

THIS ROUTINE IS USED TO DIAGNOSE BASIC FILE ERROR CODES. "ISTATUS" IS THE INPUT VARIABLE CONTAINING THE ERROR STATUS CODE. "NOMATCH" IS A RETURN VARIABLE THAT, IF EQUAL TO 1 INDICATES THAT THE ERROR IS UNKNOWN AND THE USER SHOULD CONSULT THE SYSTEM OPERATOR. IF "NOMATCH" DOES NOT EQUAL 1 THEN THE APPROPRIATE ERROR MESSAGE IS PRINTED ON THE USER'S CONSOLE.

FILERR(ISTAT, NOMATCH)





SUBROUTINE NAME.FLC

SYNTAX: CALL NAME(I)

THIS ROUTINE REQUESTS THE USERS FILE NAME AND THEN RECONSTRUCTS IT INTO A SUITABLE FORM FOR A FEASIL FILE NAME. THE FILE NAME CAN BE A MAXIMUM OF FORTY-TWO (42) CHARACTERS LONG AND THE FIRST CHARACTER MUST BE ALPHABETIC. THE NAME(I) IS RETURNED IN AN INTEGER ARRAY FORMAT WITH A MAXIMUM OF THIRTEEN ELEMENTS. THE ELEMENTS ARE:

I(1)-NUMBER OF CHARACTERS IN THE NAME.

I(2)-THE FIRST 4 CHARACTERS OF THE FILE NAME.

I(I)-THE SECOND 4 CHARACTERS OF THE FILE NAME.

. . . . .  
. . . . .

I(13)-THE TWELFTH 4 CHARACTERS OF THE FILE NAME.

NAME (I)

START

DEFINE  
VARIABLES AND  
COMMON BLOCKS

GET @,NEWWD,  
ENDFIL

SIRLEN (NEWWD,  
NAMFIL,  
LEN,  
FDIM)

IF (LEN.GE.42) F

\* 201

GETCH (NAMFIL(L2),  
1,CH)

ITYPE =  
CHTYP(CH)

IF (ITYPE.NE.1) F

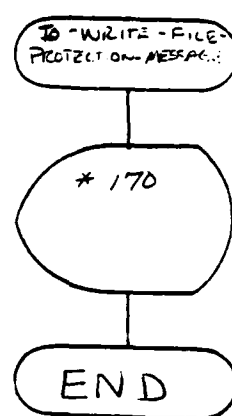
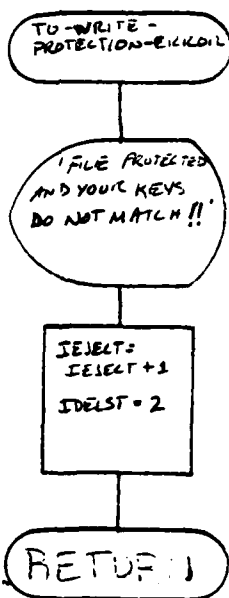
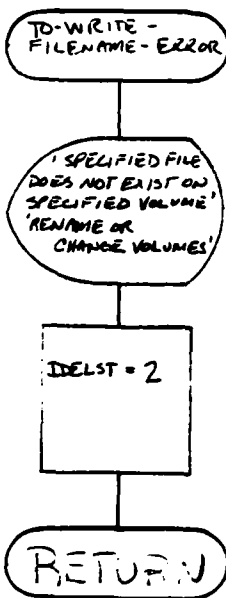
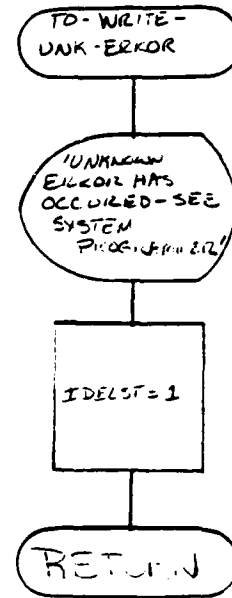
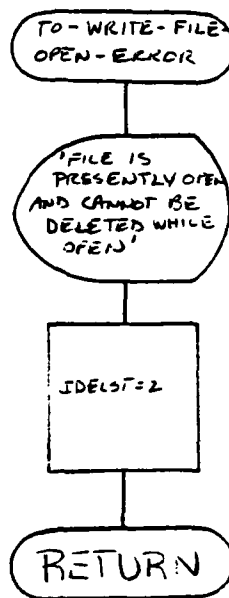
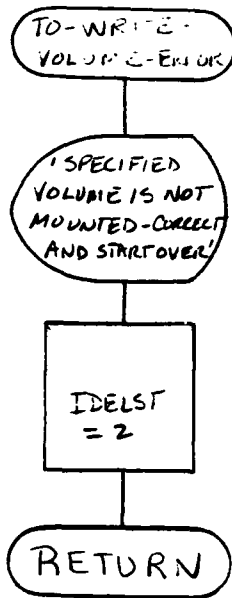
! FIRST LETTER  
MUST BE  
ALPHABETIC -  
TRY AGAIN!

EQUIV (I,NAMFIL,  
13)

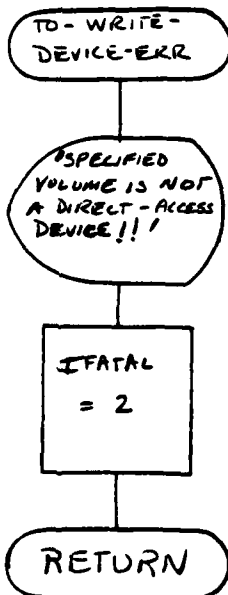
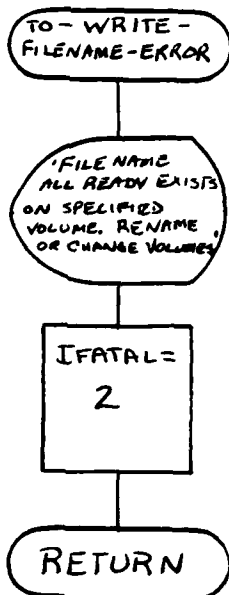
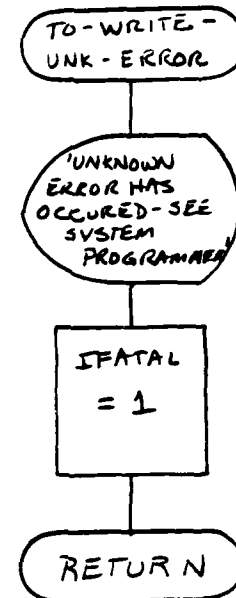
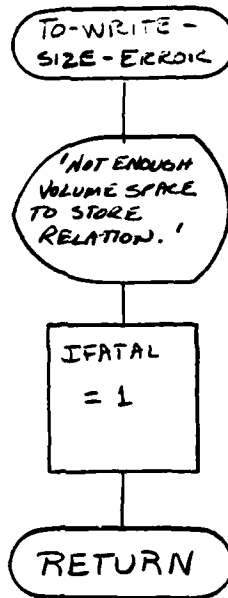
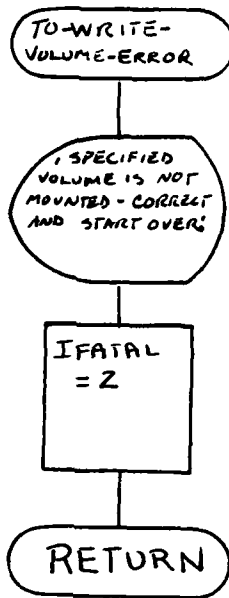
RETURN

\* 201 'NAME TOO LONG -  
42 CHARACTERS MAX -  
MUST NOT BE LONGER THAN THIS!'

TPFILECR  
PROCEDURES ASSOCIATED  
WITH SUBROUTINE RELFD  
PAGE 10



\*170 'YOU HAVE HAD 3 TRIES  
TO DELETE A PROTECTED FILE.  
YOU OBVIOUSLY DO NOT KNOW THE  
PROPER KEYS. FIND THEM  
AND TRY AGAIN LATEX.



APPENDIX T

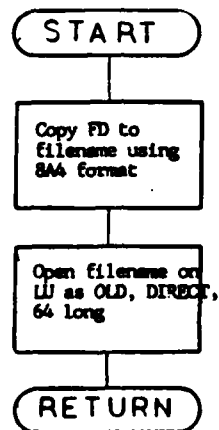
F710

Subroutine: F7IO

This routine is a collection of machine dependant subroutines that are concerned with file manipulation (opening, closing, copying, etc.) Those shown here are used on the VAX (FAST). The subroutines are:

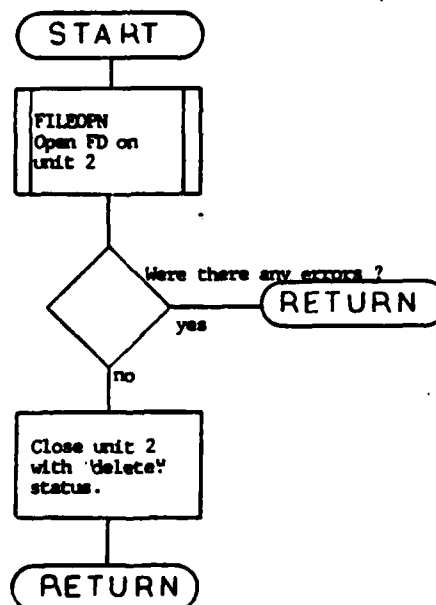
FILEOPN- opens a file  
FILEDEL- deletes a file  
CREAFIL- creates a file  
FILEREN- renames a file  
FILECLS- closes a file  
VAXERR - gives the meaning of VAX error codes.

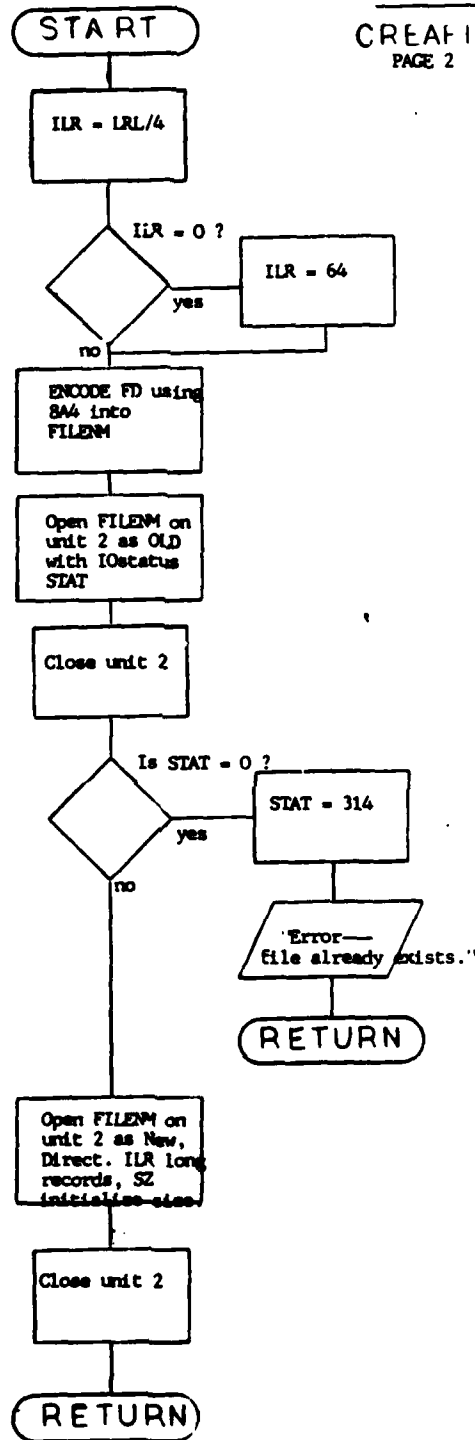
F710  
FILEOPN  
PAGE 1



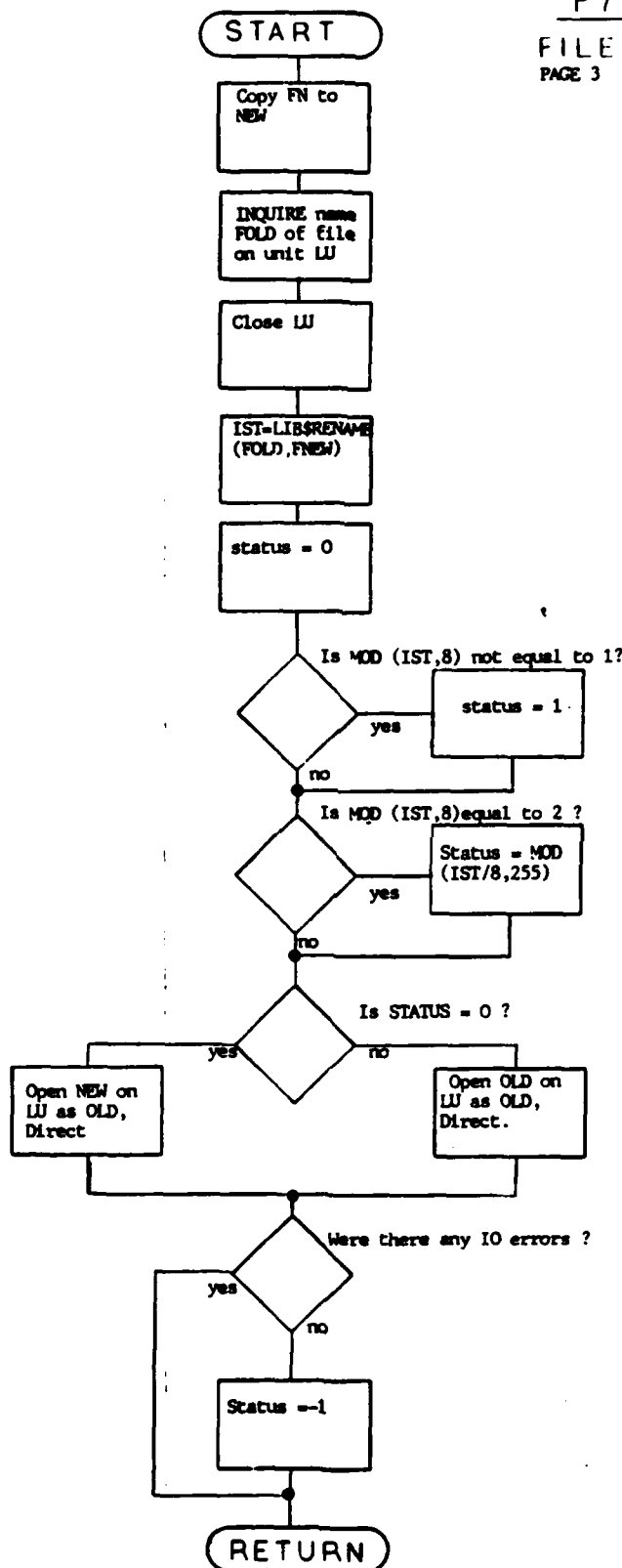
---

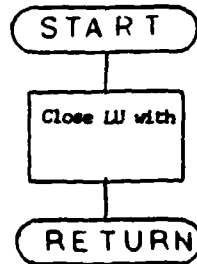
FILEDEL



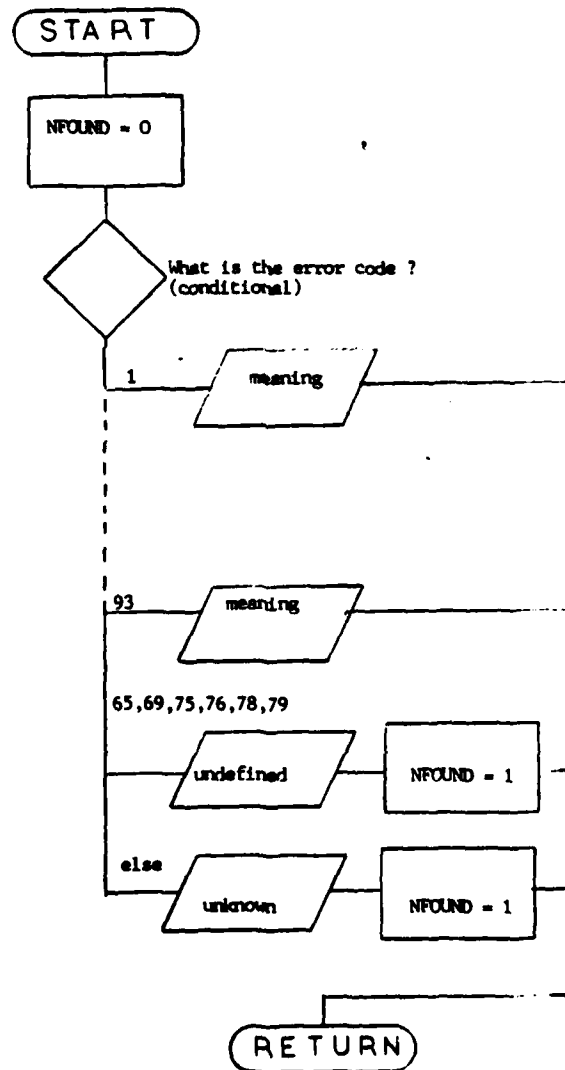








VAXERR



# DISTRIBUTION

	<u>Copies</u>
ITT Research Institute ATTN: GACIAC 10 West 35th Street Chicago, IL 60616	1
US Army Materiel System Analysis Activity ATTN: AMXSY-MP Aberdeen Proving Ground, MD 21005	1
AMSMI-RD, Dr. McCorkle	1
Dr. Rhoades	1
-RD-SS-SE, Dr. J. D. Marr	1
B. E. Tucker	5
T. A. Palmer	1
D. C. Pool	1
T. N. Long	1
-RD-CS-R	15
-RD-CS-T	1
-RD-GC-IP, Fred Bush	1

END

5-87

DTIC